AN ADAPTIVE MULTI-DIMENSIONAL SPACE MODEL FOR VISUAL RECOGNITION USING VERBAL CONCEPTS

By

A. OLABAMBO TOKUTA

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1984

Copyright 1984
by
A. Olabambo Tokuta

To my parents,Linda and Olusheni,
for their love,
support and encouragement;
To Linda and 'Sheni
for coping
with the long hours

ACKNOWLEDGEMENTS

I am indebted to a lot of people most notably my family for making this effort possible.

I wish to record my sincere gratitude to my advisor, Dr. John Staudhammer, for making this work possible and for the many hours spent guiding this research. His guidance, support, friendliness, and concern are unparalleled. It goes without saying--You're the best. Many thanks. Also, to Drs. E. Chenette, Oliver, and members of my committee for their concern, encouragement.

Finally, I cannot begin to express my appreciation to my wife Linda for her support and love, and to the tremendous amount of effort she put into the proofreading and editing of the text.

I dedicate this work to my family whose precious love and understanding have seen me through the years. Thanks.

TABLE OF CONTENTS

PAGE

ACKNOWLEDGEMENTS......................................iv

ABSTRACT...vii

CHAPTER

1 INTRODUCTION...1

2 A CONCEPTUAL THEORY....................................6

3 THE ATN FORMALISM......................................9
Augmenting the Network..................................10
Parsing...18
Formal Properties of ATN................................18
Registers in ATN..19
Ordering of Conditions and Actions......................28
Hold Registers and Locality of Context..................32
On Implementation.......................................49

4 DESIGN CONSIDERATIONS AND AN ATN BASED SYSTEM.........55
Computerized Implementations............................55
Issues..59
Form of assigned structures.............................61
Ambiguity...64
Syntactic Coverage......................................68
Specificity of Domain...................................71
System Engineering......................................72
An ATN System...73
Knowledge Representation and Semantic Networks..........80
Semantic Networks.......................................84
Conceptual Dependency and Concepts......................88

5 A MULTI-DIMENSIONAL CONCEPT SPACE.....................93
Basis and Dimension.....................................93
Concept Space...94
Objects...96
The Verbal Transformation..............................101
The Visual Transformation..............................103
Conceptual Structure...................................105

6 IMPLEMENTATION..106
The Parser...108
Representation of Meaning..............................116
An Example...121
Graphics Display System................................126

7 FUTURE CONSIDERATIONS....................127
Question Answering....................127
Declaratives....................129
Non-visual Dimensions....................130
Speech Recognition....................131
Computer Aided Instruction....................132
Conclusion....................132

8 RESULTS....................134

GLOSSARY....................146

APPENDICES

A TRANSFORMATIONAL GRAMMAR....................149

B PRONOUNS and MORE TRANSFORMATIONS....................198

C ATN PROCEDURES AND A GRAMMAR....................206

REFERENCES....................241

BIOGRAPHICAL SKETCH....................245

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AN ADAPTIVE MULTI-DIMENSIONAL SPACE MODEL FOR VISUAL RECOGNITION USING VERBAL CONCEPTS

By

A. OLABAMBO TOKUTA

APRIL 1984

Chairman: Professor John Staudhammer
Cochairman: Professor Eugene Chenette
Major Department: Electrical Engineering

The definition and manipulation of complex visual scenes are a continuing problem in computer-aided design, computer-aided manufacture (CAD/CAM) and in computer generated imagery. Usually the human-to-computer interaction is done with a specialized set of commands, an arcane set of mnemonics or a specialized language. In all these cases extensive training is required for effective image generation; also novice users are often scared away from using computed imagery for aiding designs. This treatise describes a method of generating and handling of computer generated imagery; the method is embodied in a computer program which accepts a natural language character stream as its input and produces actions on the computer display screen. The images are handled as points in a multi-dimensional concept space, and natural language descriptions of three-dimensional solid objects are displayed as visual images by the computer's display subsystem. This process forms the basis of a linguistic model for graphic entities. An object or a grouping of objects is mapped as a point or a set of points in a multi-dimensional space and properties of the object are changes in the coordinates in that space. In this system, new concepts may be input in a conversational manner based upon combinations of previously defined concepts. These new concepts are then reevaluated and may be subsequently used to form higher level concepts at the operator's discretion. New OBJECT concepts exist as a hierarchical structure of points in a multi-dimensional feature space, based upon more primitive OBJECTS. MODIFIERS are used to change their appearance by the vector displacements of these points. The concept space model is a quantitative medium for the representation of visual and verbal concepts and is compatible with accepted qualitative linguistic theories.

A portion of the model is implemented as a natural language based computer graphics language. This program uses a 'Structured' parser which incorporates a flexible strategy to transform typed English commands into the appropriate multi-space conceptualization for direct execution by a computer graphics system. A sample dialogue with the program is also discussed.

CHAPTER 1

INTRODUCTION

The study of graphics may be characterized as being that of information transfer via a visual medium, while that of linguistics may be characterized as being the study of information exchange via a verbal medium.

Most of the recent work in the field of computational linguistics has centered around the problem of language analysis, and several authors have proposed ingenious models for utilizing "cognitive" knowledge in the analysis process [WI72,83; SH73; QU67,69]. Language can be viewed as a communication process based on knowledge. In any given situation, the producer and comprehender are processing information, making use of their knowledge of the language and of the topics of conversation. Linguists have tried to understand the organization of these processes and the structure of the knowledge. Our conceptual framework is that of computation, as we understand it from our experience with stored program digital computers. The science of computation deals with theories of processing, while linguistics has typically dealt with characterization of structures. By developing idealized theories of the process, we will have a clear and revealing way of explaining the structure of their results. The computer shares with the human mind the ability to manipulate symbols and carry out complex processes that include making decisions on the basis of stored knowledge. Language is a process of communication between intelligent active processors, in which both the producer and comprehender perform complex cognitive operations. The producer begins with communicative goals, including effects to be achieved, information to be conveyed, and attitudes to be expressed. These include such things as causing an action, either verbal or non-verbal, on the part of the comprehender; causing the comprehender to make inferences or have reactions, either about the subject matter or about the interaction between producer and comprehender; conveying information about something assumed to be known to the comprehender; getting the comprehender to be aware of some new thing known to the producer; and directing the comprehender's attention to something or some of its properties, to establish a context for subsequent utterance. Some of these goals are subgoals of others. For example an utterance might have to establish a reference to an object in order to state a fact about it [WI83]. Some goals are at least partially dependent.

The details of language structure and the nature of the process involved in its use are greatly affected by the need to transmit meaning through these media. Except for a small amount of information conveyed in tone or vocal gesture, the message is forced into a linear form. There are a variety of mechanisms that make it possible to merge multiple messages into a single structure that enables the comprehender to perform the inverse process, inferring the original goals and messages from the information received.

The communicative process involves simultaneous related activities by the producer and comprehender. The crux of the model for either consist of three structures representing writing or sound patterns, syntactic structures, and representation--beliefs about the world. This model is tied to the world in two ways: through linguistic interactions and through action and perception. The basic assumption of the model, which is shared by most work in artificial intelligence, is that there are representations that operate in mental processes and that can be formally described as data structures like those of a computer. The general idea of a process operating with a knowledge base and acting to generate structures and interact with an enviroment is central to the science of computing, and its the framework for linguistic description used in the computational paradigm. In order to communicate, the producer must map this multidimensional collection of goals onto a sequence of sounds that can be uttered or marks that can be drawn on a page. In order for a computer program to "understand" an English sentence, the program must contain a model of the language which allows the meanings of individual words to be determined and somehow aggregated into the meaning of the sentence as a whole. Some linguists have broken this process down into three levels of understanding: syntactic, semantic and conceptual. The process of assigning a syntactic structure to a sentence is called parsing. The problem of parsing--recognizing sentences and assigning them structures --can be dealt with as a syntactic problem. Figure 1-1 illustrates the relationship between stored knowledge and assigned structures in the use of a parser. A parser uses a parsing algorithm, along with a grammar and dictionary, to produce a phrase structure tree that corresponds to a given sequence of words. Our problem will be one of implementing a small set of capabilities in a "large world" as opposed to implementing a large set of capabilities in a "small world" as most researchers have chosen [WI72]. While the smallness of these "worlds" would be disturbing to the computational linguist who is interested in the long range ideal, small worlds do exist in which a simple discourse can be very beneficial. Such is the world of computer graphics. It is especially suited to natural language because the depth of understanding exhibited by the computer is immediately apparent by the visual feedback generated. Since language probably evolved as a medium for describing things people saw, a model which relates visual and verbal stimuli would be an ideal kernel from which to evolve models relating other basic senses, and hence more abstract senses such as emotion.

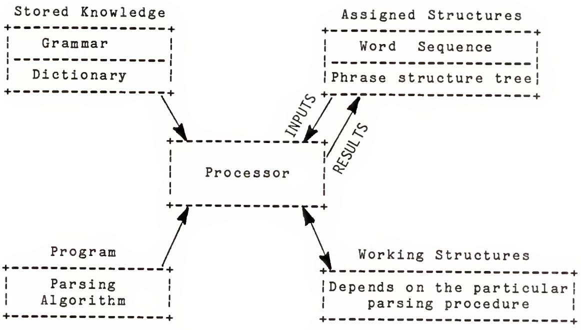


Figure 1-1. Stored knowledge and assigned structures in parsing.

CHAPTER 2

A CONCEPTUAL THEORY

The incorporation of "cognitive" knowledge into language analysis is similar to that of scene analysis and the modeling of three-dimensional objects. Thus, for visual and verbal communications we can discuss what we have done, decide what to do next, and even teach our methods to another person (or a program):

1. Concepts, both verbal and visual, are stored in the brain and may be modeled as structures in a multi-dimensional space.
2. Communication about these concepts must convey information about all aspects of these concepts; since there are a great many such aspects, reliable communication over a band-limited channel (such as a keyboard) must achieve a dimensionality reduction.
3. Transmitted information is reconstructed into its correct dimensionality model in the minds of the comprehenders; this requires that their models of the world be very nearly identical to that of the producer. Ambiguity (multiple valued inverse), anomality (null valued inverse), or incorrectness (different inverse) results when insufficient information is transmitted to uniquely reconstruct the correct multi-dimensional model.

In the past twenty-five years a new paradigm has gained predominance largely due to Chomsky [CH57]. He argues that the proper domain of study, according to generative paradigm, is not the sentences themselves but the underlying faculty that enables us to create and understand them. He proposed that theory should deal with relations between structures by looking both at surface structures, which had been studied previously and deep structures which capture the underlying similarities and differences in his model of language structure called Transformational Grammar. In this grammar, sentences are seen as "surface structures" which represent, at a conversational level, kernels of information contained in a "deep structure." The transformation from surface to deep structure is many-to-one so that a single deep structure can generate or be generated by a number of different surface structures. Verbal communication is thus modeled as (1) a deep structure conceived by a producer and (2) transformed into a surface structure which is then articulated. Upon receipt of the surface structure (3) the comprehender performs an inverse transformation to determine (4) the deep structure transmitted.

These structures are related to one another by grammar where a grammar may be formally defined as a series of generalized rewrite rules of the form

$$\Theta \dashrightarrow \Phi$$

which are a formal link between the simpler grammars and the mechanisms of transformational grammar (TG). Any symbol Θ appearing in the structure may be replaced by Φ repeatedly until there are no more in the string.

The nature of the deep structure has not been mathematically formalized, although linguists have developed many symbolic models which use it. It is proposed that deep structure may be modeled as structures in a multi-dimensional space and that transformations may be defined mathematically.

CHAPTER 3

THE ATN FORMALISM

In the period that TG emerged, development in AI followed a different approach. Rather than taking the process of abstract derivation as a starting point, they chose the process of parsing. The current paradigm of linguistic theory, transformational grammar, is not very suitable for either theoretical or applied computational purposes. Augmented transition networks (ATN), Appendix C, provide a model which captures many of the ideas of transformational grammar in a computational efficient and theoretically interesting way [BA81]. ATN and their descendants are currently among the most common methods of parsing natural language in computer systems. They have also served as a basis for psycholinguistic theories and experiments.

They have been used in understanding systems and question answering systems for both text and speech. They have proved to be flexible, and able to handle a wide variety of syntactic constructions, and easy to interface to other components of a total system. ATN grammars in its popular form were developed by Woods [WO70]. The advantages may be summarized as (1) perspicuity, (2) generative power, (3) efficiency of representation, (4) ability to capture linguistic regularities and generalities, and (5) efficiency of operation [BA81b].

A transition network is used in place of rules(which represent a possible structure for a phrase) to define a language, either by itself or as one of a set of networks in an ATN grammar. A network consists of a collection of finite state transition diagrams. Each is a directed graph with labeled states and labeled arcs. States can be designated as initial and terminal states. Each arc is labeled with a word, a lexical category, or a syntactic category that is the label of some network in the grammar. A recursive transition network grammar contains a network for each non-terminal symbol of the grammar. Arcs are assigned an order so that a backtrack strategy can be more efficient by trying the more likely ones first. ATN allows some special kinds of arcs, one of which (send arc) is used as the terminal state. A network is not used to store information about a particular parsing. It represents a pattern to be matched against potential sentences. The ATN formalism is oriented towards top-down, left-to-right (TLR) parsing. The setting of registers and the use of their contents in specifying conditions must take place in the proper order, or the results will be totally garbled.

Augmenting the Network

Recursive transition networks have the same formal power as CF grammars but are more convenient form to work with. Their augmentation enables the handling of the more complex syntactic phenomena. It is important to know that many of the generalizations captured by the theory of transformational grammar can be handled smoothly by an ATN grammar. The augmentation centers around the "conditions" and "actions" associated with the arcs of a network. Conditions restrict the circumstances under which an arc may be taken, while actions perform feature-marking, and structure-building operations. These make use of registers for roles and features, associated with the nodes of the parse tree being constructed.

Conditions may depend on constituents that have already been found, on special properties of the word or constituent to be matched against the arc, and on properties of the entire structure that has been built up so far. Actions are used both to generate the ultimate structures that result from analysis and to provide temporary information that can be used by conditions on other arcs.

Register Structures

Woods [WO70] original ATN formalism used more standard tree structures. Here register structures are not trees but a more general graph structures, since there are instances that call for sharing of component (as in embedded sentences). We present a kind of syntactic structure different from the phrase structure trees but more closely related to the role structure trees. Role structure trees are similar to phase structure trees but use role(register) table which associates role names with other nodes, sequences of nodes, or symbols. Each key in the table is a slot and the corresponding entry its filler. This method attributed to Kaplan [KA75] and adapted from Winograd [WI83], incorporates register structures. Register structures are more general graph structures, not trees--so as to allow sharing of contents. The way registers are typically used forces a strong orientation to TLR processing.

A register structure (Figure 3-1) is made up of nodes, each with a label (as in p-structure trees) and a register table (Figure 3-2). A register table associates feature dimensions and role names with particular values. The entry for a feature dimension is a feature chosen from the possible values for that dimension, and the entry for a role is a node or a sequence of nodes. In Figure 3-1, each node is represented by a box and the roles are indicated by arrows to the appropriate other boxes. In addition to having the extra feature markings and role labels, these structures differ from tree structures in that no ordering is specified for the role fillers and the same node may indicate more that one role. As an example, the feature dimension NUMBER has choices PLURAL/SINGULAR while the role SUBJECT is filled by a NP node and the role DESCRIBERS by a sequence of ADJECTIVE nodes.

```
+-------------------++------------------------+     +------------+
|S|We have...school||NP|the graduate school| +->|Det| the    |
|------------------||------------------------| |   |------------|
|Mood: Declarative ||Number: Singular       || |   |Number: Sing|
|Voice: Passive    ||Person: 3rd            || |   |        Plu |
|                  ||------------------------| |   +------------+
|                  ||Determiner          o---+     +------------+
|                  ||Describers          o------->|Adj|Graduate|
|Subject       o--->Head                o---      +------------+
|                  |+------------------------+ |
|                  |                           |   +------------+
|                  |                           |   |Noun|school |
|                  |                           +-->|------------|
|                  |                               |Number: Sing|
|                  |                               +------------+
|                  |                               +------------+
|                  |                               |Verb | have |
|                  |                               |------------|
|                  |                               |Form: Pres  |
|Auxiliaries    o------------------------------+-->|Type: have  |
|                  | +----------------------+ |   |Trans:Trans |
|                  | |Verb |  given         | |   +------------+
|                  | |----------------------| |
|                  | |Form: Past-Part       | |   +------------+
|                  | |Type: Non-aux         | |   |Verb | been |
| Main Verb       o-->|Trans: Bitransitive   | |   |------------|
|                  | +----------------------+ +-->|Form:Past-P |
|                  |                               |Type: Be    |
|                  |                               |Trans:Trans |
|                  |                               +------------+
|                  |
~                  ~
~                  ~
```

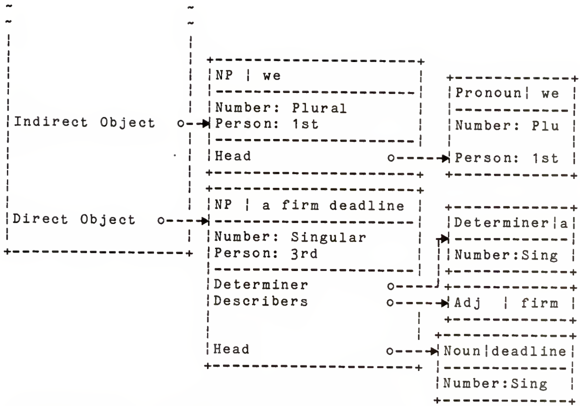


Figure 3-1a. A sample register structure
"We have been given a firm deadline by the graduate school."

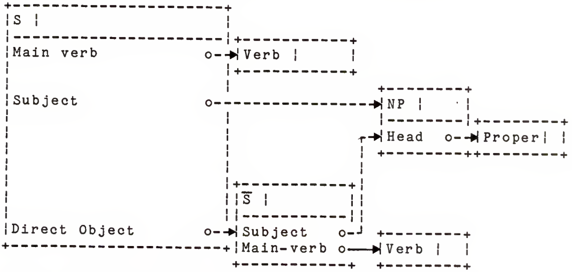


Figure 3-1b. Example of a structure including shared component.

Figure 3-1. Register structures.

```
+----------------------------------------------------------+
!                         NODE                             !
+----------------------------------------------------------+
 Describes: a node created in parsing a constituent
 Background: an ATN grammar and a dictionary
 Roles:
    Label: a syntactic category
    Register: a register table
 Classes:
   +----------------------------------------------------+
   !                  REGISTER TABLE                    !
   +----------------------------------------------------+
    Describes: the register contents for a node
    Structure: a table whose keys are register names
        and whose entries are:
      - If the key is the name of a feature dimension,
           then a choice from that dimension.
      - If the key is one of the role names of a network,
           then either a node or a sequence of nodes.
     Every register table is associated with a particular
     node and with a single network or lexical category
     that is the label of that node.
   +----------------------------------------------------+
   !                  REGISTER NAME                     !
   +----------------------------------------------------+
    Describes: the key for an entry in a regiter table
    Structure: a sequence of characters
     A register name has no relevant internal structure.
     All that is needed is that it is distinct from
     others associated with the same network or lexical
     category. The name is either a role name from a
     network or the name of a feature dimension.

+----------------------------------------------------------+
```

Figure 3-2. Register structure node

Associated with each network is a set of feature dimensions and a set of role names. When a phrase is parsed, the register table of the resulting node contains entries whose keys are the feature dimensions and role names for the network named in its label. Features and set of features that can be accessed are

(i) Values given explicitly in an action or condition.

(ii) The contents of a feature register of the current configuration.

(iii) The contents of a feature register of some accessible node.

Nodes that can be accessed are

(i) The contents of a role register of the current configuration.

(ii) The most recently parsed constituent.

(iii) The contents of a role register of some accessible node.

(iv) The first or last element of some accessible sequence.

(v) Dummy nodes created explicitly in actions.

(vi) A copy of the current node (used in relative clauses).

Sequences of nodes that can be accessed are

(i) The contents of a role register of the current configuration.

(ii) The contents of a role register of some accessible node.

Initializations. When an arc is traversed that calls for parsing with a network; some of its registers can be initialized to contents contents derived from the status of the parsing so far. Initializations consist of

(i) A key, which is the name of a register in the network being initialized.

(ii) A value, which accesses features or nodes in the configuration from which initialization is being done.

Conditions and Actions. Conditions and actions are associated with the arcs of the grammar. For an arc to be traversed, its condition must hold for the current state of the parsing, along with the normal arc-applicability requirements. The associated action causes the contents of the register to be set. Actions that can be taken include:

(i) A feature register is set to a choice for its dimension.

(ii) A role register is set to a node.

(iii) A node is appended to the end of a role register containing a sequence of nodes.

Conditions on accessible values that can be tested include:

(i) Two features are identical.

(ii) A role register is empty.

(iii) A node is of a specified category.

(iv) A node is a dummy node (created by a specified action).

(v) The word associated with a node is a specified word.

(vi) Any logical combination (or, not, if . . .then, etc.) of the above.

Classification of arcs. Arcs are classified (Appendix C) as a "category" arc (CAT) which is matched against a single word of the input. Its label is a lexical category. A "seek" (PUSH) arc specifies a recursive call to a network (possibly the one in which it appears). Its label is that of the network or of a state in it. A "send" (POP) arc which has a starting but no ending state. Thus it can thought of as a terminal state of the network, its effect is to register a successful recognition of that network. They enable easy association of actions and conditions with the successful recognition of that network, since they can be attached to a send arc just as to any arc. "JUMP" arcs are taken without parsing an element of the input string. They often simplify the writing of networks, and can have associated conditions and actions.

Parsing

The schema for parsing with an ATN grammar is shown in Appendix C. Actions put contents into a register and the conditions constrain which arcs can be taken. A temporary structure which is a representation of the state of an ATN parse is used. A configuration includes the register table for the constituent being built, together with a current network, state, position, and the node most recently parsed. A node is not actually built until the send arc is taken at which time the register table of the configuration is incorporated into the node.

By assemblying all of the information into a single structure it is easy to use backtracking or parallel schemes. A parallel scheme would keep track of a number of alternative configurations simultaneously while a backtracking scheme would save configurations in order to possibly return to them.

Formal properties of ATNs

ATN formalism can be described and shown equivalent in power to a Turing machine [BA78]. Woods has also shown that it is possible to eliminate direct left and right recursions and to optimize a network using finite state optimization techniques. Earley's algorithm for CF recognition [EA70] follows alternatives in parallel, using a particularly clever method of merging parse paths. It can parse strings in an amount of time bounded by (K)*n**3 on an input of length n (with K a constant of proportinality dependent on the grammar and not the input). This algorithm works for CF grammar in any form (not just normalized) and it automatically achieves the smaller bounds for linear or LR(k) grammar without being told that the grammar is so restricted. For linear grammars, the recognition time is bounded by n**2, n for LR(k) grammar. An algorithm due Valiant[VA75] is asymptotically and in the worst case more efficient than Earley's; but more complicated to implement takes time proportional to n**2.81.

By modification of Earley's algorithm, ATN grammar can be recognized to maintain the time bounds. By removing recursion from an ATN causes the upper time bounds to be reduced from n**3 to n**2 or n. Simple optimization may be used to reduce the constant of proportionality in the n**3 case.

Registers in ATN

In transformational grammar, the solution to the syntactic problem of number agreement was handled by the addition of grammatical markers (SINGULAR/PLURAL) which was subsequently manipulated by transformations; or the use of features associated with individual lexical items. Here we allow features with multiple choices and associate feature registers with networks as well as features on individual words. Use of features is based on a formal structure called feature dimension. Each dimension has a name, a set of choices and an initial default value(Figure. 3-3).

Category	Dimension	Choices	Default
Adjective	-		
Complementizer	-		
Determiner	Number	Singular, Plural	Singular, Plural
	Question	Yes, No	No
Noun	Number	Singular, Plural	Singular
	Case	Subjective, Objective	Subjective, Objective
Preposition	-		
Pronoun	Number	Singular, Plural	Singular
	Person	1st, 2nd, 3rd	3rd
	Case	Subjective, Objective	Subjective, Objective
	Question	Yes, No	No
Proper	Number	Singular, Plural	Singular
	Case	Subjective, Objective	Subjective, Objective
Relative	-		
Verb	Form	Infinitive, Present, Past, Past-Participle, 3rd-Singular-Pres, Present-Participle	Infinitive, Present
	Transitivity	Intransitive, Bitransitive, Transitive	Transitive
	Type	Be, Do, Have, Modal, Non-Aux	Non-Aux

Additional verb dimensions would be needed to handle all forms of "be."

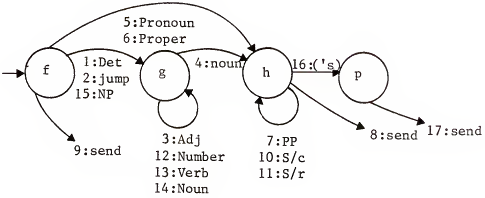
Figure 3-3. Examples of lexical categories.

Feature Registers. Figure 3-4 illustrates sample dictionary entries based on categories of Figure 3-3. Each word is associated with set of lexical categories to which it belongs and for each lexical category, a set of feature dimensions.

Sample word	Category	Features
a	Determiner	Number: Singular
been	Verb	Form: Past-Part, Type: Be, Non-aux
box	Noun	
boxes	Noun	Number: Plural
by	Preposition	
chicken	Noun	Number: Singular, Plural
	Verb	Transitivity: Intrans
	Adjective	
deadline	Noun	
firm	Adjective, Noun	
fish	Noun	Number: Singular, Plural
	Verb	Transitivity: Intrans
	Adjective	
given	Verb	Form: Past-Part, Transitivity: Bitrans
have	Verb	Type: Have, Non-aux
having	Verb	Form: Pres-Part; Type: Have, Non-Aux
John	Proper	
man	Noun	Number: Singular
	Verb	Transitivity: Trans
	Adjective	
saw	Verb	Form: Past
	Noun	Number: Singular
secretary	Noun	
that	Complementizer	
	Determiner	
	Pronoun	
	Relative	
the	Determiner	
these	Determiner	Number: Plural
this	Determiner	Number: Singular
to	Complementizer	
	Preposition	
want	Verb	Transitivity: Trans, Bitrans
we	Pronoun	Number: Plural; Person: 1st; Case: Subjective
were	Verb	Form: Past; Type: Be, Non-Aux
which	Determiner	Question: Yes
	Relative	
who	Pronoun	Case: Subjective; Question: Yes
	Relative	
whom	Pronoun	Case: Objective; Question: Yes
	Relative	

Figure 3-4. Sample dictionary entries.

A word can have several word senses, each of which is assigned to a category and a choices along each of the feature dimensions for the category. The same word may belong to different categories and have different features in each. The default value for each dimension is assumed to hold for any word sense for which a value in that dimension is not specified. When more than default is given, there is a word sense for each. For example a verb that does not specify FORM is assumed to have one sense for INFINITIVE and one for PRESENT. We illustrate number agreement in the NP network, with feature dimensions for nouns and determiners as in Figure 3-3 and introduce actions (A) and conditions (C) on to the arcs to check number agreement between noun and determiner (Figure 3-5). The actions and conditions associated with arcs for modifiers such as adjective and PPs will not be considered since they do not affect number agreement in English as in many other languages.



```
Feature Dimensions: Number: Singular, Plural;
                    default, -empty-

Intializations, Conditions, and Actions:

NP_f1g
        A: Num <- (#)num.
NP_g4h
        C: Number is empty or Num = (#)num.
        A: Num <- (#)num.
NP_f5h
        A: Num <- (#)num.
NP_f6h
        A: Num <- (#)num.
```

Figure 3-5. NP network and number agreement

This will handle noun phrases provided there exist choice of word senses for the individual words that allow the conditions to be satisfied. It will fail for ungrammatical phrase like "these box." In case of words with multiple entries (e.g. the) a phrase that calls for a particular choice, "the box," will succeed with some choices and fail with others. Since ATN formalism is nondeterministic, a phrase is grammatical as long as one successful combination of choice exist. In ambiguous cases like "the fish," there will be more than one potential successful result. If the NP consist of a pronoun, or a proper noun, the number feature of the NP as a whole will be that of the word.

This figure and that in Appendix C illustrate aspects of the ATN formalism: In the latter, the jump arc does not have any conditions or actions associated with it. The send arc allows conditions and actions to be associated with the completion of a network. The remaining--category arcs--specify the lexical category of the word to be matched. This involves the creation of new nodes whose contents are taken from the appropriate word sense for the word as in Figure 3-4. Associated with the network is a feature dimension for NUMBER. Each of the node has an entry in its register table containing a choice of SINGULAR/PLURAL. Here, the same dimension also apply to individual words. Feature dimension are also applicable to networks and do not appear in the dictionary. Following the name of a feature dimension is a list of all the possible choices, followed by the default, which is the initial setting of the register.

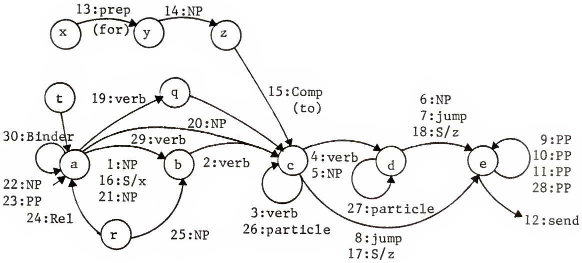
A kind of action that can be associated with an arc is to set the choice in some feature register. Oftentimes, action on an arc refer to node that was created in a match. Where the case is such that the label of the arc is a lexical category, this will be the node whose label is the lexical category and whose feature choices are those in the corresponding part of the dictionary entry for the word matched by the arc. When the arc is a seek arc, it will be the node whose label is the name of a network and whose contents are the registers for that network. The symbol # would refer to the entire node. Thus, the action "Set NUMBER to . . ." (Num <- (#)Num) specifies the setting of that register, for NUMBER, to the given value.

To refer to a particular set of features, the name of the feature dimension is used--"the Number of #." The contents of the corresponding register of the node are used.

Therefore when the NP consist of a pronoun (NP_f5h) or a proper noun (NP_f6h), the number feature of the NP is that of the word.

Role Registers. The network of Figure 3-6 manipulates register contents so that the DIRECT-OBJECT register will end up containing the "deep direct object" regardless of whether the sentence is active or passive. The subject register will contain the deep subject--if given in the sentence--or a dummy NP if the sentence is an agent-deleted passive. We associate a SUBJECT register and a DIRECT-OBJECT register with the network and have them assigned by the action as shown.

Associated with each network is a set of role names. Part of the structure created in parsing is the register table, which contain both the feature and role registers. These are initialized before any arc is taken. When an assignment is made (e.g. in parsing an NP--"Set SUBJECT to # . . .") the contents will be the NP node just parsed until another assignment is made; and the old entry is replaced.



Roles: Subject, Direct-Object(d-o), Main_Verb(m-v),
Auxiliaries(aux), Modifiers(mod)

Feature Dimensions: Voice: Active, Passive; default, Active

Initializations, Conditions, Actions:

S_a1b
- A: Subj <-- #.

S_b2c
- A: m-v <-- #.

S_c3c
- A: auxs <= m-v; m-v <-- #.
- C: The type of the main-verb is Be, Do, Have, or Modal

S_c4d
- A: Voice <-- Passive; auxs <= m-v; m-v <-- #; d-o <-- subj; subj <-- dummy NP
- C: Form of # is past-part; m-v type = Be

S_c5d
- A: d-o <-- #.

S_d6d
- A: Mods <= #.

S_d7d
- A: Subj <-- Prep-Obj of #.
- C: Voice = Passive; Subj = dummy NP; Word in Prep of # = by.

S_d8
- Send: No init, cond, or actions.

Figure 3-6a. S network with active and passive voice.

```
+------------------------------------------------------------+
  Additional conditions and action:

  S_b2c
         C: Either: type of # is Modal
            or Form of # is Past
            or Form of # is 3rd-Singular-Present and Number
              of Subj is Sing and Person of Subj is 3rd
            or Form of # is Pres and either Num of Subj is
              Plural or Person of Subj is 1st or 2nd
+------------------------------------------------------------+
```

Figure 3-6b. Conditions and actions for subject-verb agreement

Figure 3-6. A S network.

A register is used for MAIN-VERB(m-v) which is assigned on three different arcs. This is initially assigned on the S_b2c, which takes the first verb, and is reassigned for each possible match on the S_c3c arc and S_c4d arc if taken. The resulting entry for the register is the last verb encountered, which is always the main verb of the sentence--in the sense of bearing the meaning of the action or relationship being described. Meanwhile, the AUXILIARIES register contains a list of the auxiliary verbs preceding the main verb.

Role registers can be appended; when there is a loop in the network it is filled by a sequence of constituents rather than by a single constituent. Auxiliaries and modifying phrase of a sentence are examples of such constituents. An action like "Append MAIN-VERB to AUXILIARIES" appends the contents of the MAIN-VERB register to the end of the previous contents of AUXILIARIES. If the register was previously empty, it is filled by a sequence consisting of just the new item.

Registers containing sequence of elements can be accessed as "the first element of. . . ." We need not access arbitrary positions within grammars in order to write reasonable grammars. This fact is related to the notion of "structure dependence" formulated for transformations.

Arcs differing only in condition and actions as with the preposition "by" may be parsed by a special PP arc, S_e10e, instead of by the arc intended for modifiers. This is necessary because there must be separate paths through the network for "by" phrases that specify agents, and those that do not, even though they appear in a passive sentence. Ambiguity will result in multiple parsing, each corresponding to a different sequence of arcs taken in matching the networks.

Conditions specified on arcs have to be met, otherwise the arc cannot be taken--the match fails. A condition can specify any Boolean combination of tests by use of "and," "not," "or," "if . . .then."

Ordering of Conditions and Actions

The conditions and actions in an ATN are designed with a specific parsing order in mind. Usually it is assumed that the parser is TLR, and hence actions that sets a register will affect conditions on arcs to the right, but not those previously traversed. This assumption is both an advantage and a limitation. It can make it possible to provide, in a relatively simple way for the interactions between different syntactic phenomena. On the other hand it limits the flexibility in the design of the parser, and necessitates a modification to the formalism to operate bottom up or not purely left-to-right fashion.

The subject-verb agreement. One motivation for multiple verb arcs in Figure 3-6 is that it provides a clear place for the agreement check since agreement always applies to the first verb regardless of whether it is the main verb or an auxiliary. Often decisions about the states and arcs of the network are determined by this kind of consideration.

Subject-verb agreement could be handled in a number of ways, as is often the case in writing the rules of the grammar. Actions and conditions to be applied are enumerated in Figure 3-5 and appendix C. Arc S_b2c has agreement condition which makes use of the SUBJECT register, and an action on S_c4d that can potentially change its contents. Clearly, the condition needs to apply to the subject before the change. The sentence "Some apples were . . . by a giraffe" has plural surface subject "Some apples" agree with the verb "were" and not the deep subject "a giraffe." This happens naturally, since the arcs are taken in the left-to-right sequence indicated in the network. This kind of phenomena are closely related to the rule ordering of transformational grammar, but are more natural, since the order is imposed by the order of the phrases in the sentence rather than by a less intuitively guided choice of ordering for derivation rules.

Transitivity. Associated with the lexical category of a VERB is a feature dimension TRANSITIVITY. The choices are TRANSITIVE, INTRANSITIVE, BITRANSITIVE according as to whether there are one, none, or two objects respectively. Transitivity is a much more complex phenomena which is not fully handled in any grammar formalism. Again, we add some states and arcs to the S network(Figure 3-7) and accommodate an intervening state. The network uses jump arcs to account for the different possible combinations of transitivity, voice and datives.

In the active case, after the main verb is found, the S_c5d arc can be taken to find the first object, or the jump arc S_c8e can be taken to skip past finding of objects. Once the first object is found, the S_d6e will find a possible second one while S_d7e accounts for the nonexistence.

Arc S_c5d stores the first object found, if there is one, as the DIRECT-OBJECT and remains there if the S_d7e arc is taken. If a second comes along, the first is moved by arc S_d6e to INDIRECT-OBJECT and the new one becomes the DIRECT-OBJECT.

The network will parse as many objects as can be found, even if the transitivity features on the verb do not allow them. The check for appropriate transitivity is done on the final send arc, after all the arcs have been taken, thus allowing the conditions to be stated all together.

Here again, we recognize the importance of ordering. Since transitivity is checked for after the other arcs, an

```
+-------------------------------------------------------------+
  Roles: Subj, d-o, i-o, m-v, auxs, mods.
  Feature Dimensions: Voice: Active, Passive; default,active
+-------------------------------------------------------------+
  Initializations, Conditions, Actions:

  Arcs 1-5: see fig. 3-6.
  S_d6e
         A:i-o <-- d-o; d-o <-- #
  S_d7e
         Jump, no actions, conditions
  S_c8e
         Jump, no actions, conditions
  S_e9e
         A: mods <= #
  S_e10e
         A: Subj <-- (#)Prep-obj.
         C: Voice = Passive; Subj = dummy NP;
            by=word in Prep of #.
  S_e11e
         A: i-o <-- (#)Prep-obj.
         C: word in Prep of # = (to or for); i-o = 0
  S_e12
         C: If i-o  ≠  0 then m-v.Trans = Bitransitive
            If i-o=0 and d-o  ≠  0 then m-v.Trans=Transitive
            If d-o = 0 then m-v.Trans = Intransitive
+-------------------------------------------------------------+
```

Figure 3-7. Actions for transitivity and dative in S network.

indirect object that is found in a PP starting with "to" or "for" will result in a failure unless the verb is bitransitive. If the check was done earlier it would fail to interact properly with this kind of dative.

Hold Registers and Locality of Context

In our discussion of transformational grammar we alluded to forms of deep structure having an embedded sentence with subject same as that of, or object of the matrix sentence. This was handled by having identical NP nodes and deleting one of them through EQUI. We now consider the handling of syntactic phenomena involving information "forced" from a higher node to one of its constituents. There are many different types of clauses to be considered each with its own distribution. Figure 3-8 is extended to handle a number cases in which both subjects and objects are embedded sentences. It permits the inclusion of specific words on category arcs: S_x13y(for) and S_z15c(to) have conditions that allow them to match only specific words--lexical(LEX) items. In some ATN implementations there is also a "MEM" arc, which specifies a list of words and is conditional on the word parsed being a member of the list.

This network also illustrates the notion of multiple entry points into a network. The state S.x is not reachable from the initial state S.a of the network. Arc S_a16x/b specifies that the network is to be matched beginning in S.x instead of S.a. It is thus possible to jump right into the middle of the network. A subject is found from the matrix

```
Roles: Subj, d-o, i-o, m-v, auxs, mods.
Feature Dimensions: Voice: Passive, Active; default,active
```

```
Initializations, conditions, actions:

Arcs 1-4: see figure 3-6; arcs 5-12: see figure 3-7

S_x13y(for)
        C: for = word in #.

S_y14z
        A: Subj <-- #. C: (#)Case = obj.

S_z15c(to)
        A: m-v = dummy verb with type = modal.
        C: to = word in #.

S_a16(s/x)b
        A: Subj <-- #.

S_c17(s/z)e
        A: d-o <-- #.
        I: Init Subj = Subj of ↑ (Subj <--↑  Subj)

S_d18(s/z)e
        A: i-o <-- d-o; d-o <-- #.
        I: Subj <-- (↑) d-o.
```

Figure 3-8. 'to' complements in S network.

sentence and put into a register before parsing with the S network begins for the embedded sentence. By joining the network in S.c after finding the "for. . .to" phrase allows the accumulated structure to be shared. The action on S_z15c(to), which sets the MAIN-VERB register to a dummy verb, causes proper agreement between adjacent verbs to be checked, for the verb following "to" must be in the INFINITIVE form as is with a verb following a modal. Since the normal verb-agreement check is on the S_b2c arc, it is bypassed in this case. In any case, it is possible to take one of the arcs out of S.c when the main verb register contains a modal. In this case the parse will fail when the transitivity check is tried at the end.

When the network is entered at S.x, the SUBJECT and MAIN-VERB registers will be set by the time S.c is reached. Consider a sentence with components NP - V - to - V. The subject in the sense of meaning is the subject of the outer sentence. The embedded sentence "to - V" is matched by the S_c12z/e arc. The initialization on the arc requires the node created by the arc, due to the embedded sentence, to have its subject register initially contain the contents of the subject register of the calling sentence--indicated by ↑. This action results in two registers set to the same contents--shared component--in which a single node fills a role in more than one node (Figure 3-2). In this case unlike previous initialization settings (defaults), the setting is done in the particular context. The subject register of the embedded sentence has its contents taken from the matrix sentence. Arc S_d18z/e handles the case where the NP is both the indirect object of the matrix and the subject of the embedded; by moving the previous direct object to the indirect object register after parsing an embedded sentence where it is initialized as the subject.

The initial register setting is a way to establish the connection of multisharing of component. We could have parsed the embedded sentence without the subject and have the action on the S.z arc insert the appropriate subject. This however, would not work properly when combined in the passive voice. When the embedded sentence is passive, the subject needs to be in place initially so that it can participate in actions that move it into the DIRECT-OBJECT register.

This example shows how a careful ordering of arcs with their initialization and action can effect the same result as the ordering of transformation and mechanisms like the transformational cycle.

Questions. The handling of questions are not as simple, since the passed-down phrase is not limited to a particular role(the subject) but can appear anywhere within the S, including within an embedded constituent to any level of depth.

The network of Figure 3-9 includes new arcs and registers to deal with different forms of questions. The feature dimension MOOD has the choices: DECLARATIVE, INTERROGATIVE, and IMPERATIVE. The state S.q with associated arcs S_a19q and S_q20c handle the "subject-auxiliary" inversion which marks the interrogative mood. The first verb of the sequence appears before the subject, rather than after it. The mood check will prevent this arc from being taken in embedded sentences like relative clauses. It will however be taken

```
+------------------------------------------------------------+

  Roles: Subj, d-o, i-o, m-v, auxs, mods,
         question-element(q-e)
  Feature Dimensions: Voice: Passive, Active; default,Active
  Mood: Declarative(dec), Interrogative(int),
        Imperative(imp); default, Declarative

+------------------------------------------------------------+

  Initializations, conditions, actions:

  Arcs 1-4: figure 3-6, arcs 5-12: figure 3-7

  Additional conditions on S_a1b:
                C: Ques of # = No; Mood ≠ int

  Additional conditions on S_e12(send)
                C: Hold = 0

  S_a19q
         A: m-v <-- #.; Mood <-- int.
         C: (#)Type ≠ Non-aux; Mood = dec or int.

  S_q20c
         A: Subj <-- #.
         C: Subj-verb agreement as in figure 3-6b.

  S_a21b
         A: Subj <-- #; q-e <-- #; Mood <-- int.
         C: (#)Ques = Yes; Mood = dec; (#)Case = subj.

  S_a22a
         A: q-e <-- #; Hold <-- #; Mood <-- int.
         C: (#)Ques = Yes; Mood = dec.

  S_a23a
         A: q-e <-- (#)Prep-obj; Hold <-- #; Mood <-- int.
         C: (#)Ques.Prep-obj = Yes; Mood = dec.
+------------------------------------------------------------+
```

Figure 3-9. S network to include questions.

following an initial question element. Once the auxiliary and subject are consumed, the rest of the sentence parsing continues as usual. The subject-verb agreement check is duplicated on the S_q20c arc since the S_b2c arc will not be taken.

Wh-questions are handled by the QUESTION-ELEMENT register. In such sentences, knowing what is questioned is a key element of the meaning and must be a part of the analysis produced. The S_a21b arc handles the case where the subject of the sentence is a QUESTION NP. The conditions on the arc and added conditions on the S_a1b arc ensures that a sentence with a QUESTION subject will be parsed as a sentence.The condition on S_a1b directs any sentence with an initial question element that is not treated as the subject to follow the path through S.q. Feature dimension labeled QUESTION(choices YES/NO) applies to NPs, pronouns and determiners. YES applies to NPs with question pronouns like "who," "what" and to those with a question determiner like "which," "how many." The mood condition on S_a21b eliminates ungrammatical forms, while corresponding conditions on S_a22a and S_a23a forbid multiple question elements caused by looping. Figure 3-10 shows the necessary extension to the NP network.

Long-distance dependencies. Long-distance dependency is one of the most complex phenomena of syntax. It has been a major motivation for many of the mechanisms of TG and the need to account for the phenomenon is a major force shaping grammar formalism. It was one of the motivations for the original idea of transformations and in some versions of TG, the only transformations remaining are those needed to handle it. Some element of the structure is constrained by features of another part of the structure that is not its parent or one of its siblings, but is located at a distant place in the tree.

```
+-----------------------------------------------------------+

  Roles: Determiner(det), Head(hd), Describers(des),
           Qualifiers(qua)
   Feature Dimensions: Number: Singular, Plural;
           default, empty
       Person: 1st, 2nd, 3rd; defualt 3rd.
       Question: Yes, No; default, No

+-----------------------------------------------------------+

  Initializations, Conditions, actions:

  NP_f1g
         A:det <-- #; Num <-- (#)Num; Ques <-- (#)Ques.
  NP_f2g
         No init, conds, or acts
  NP_g3g
         A: des <= #.
  NP_g4h
         A: hd <-- #; Num <-- (#)Num.
         C: Num = 0 or Num = (#)Num.
  NP_f5h
         A: hd <-- #; Num <-- (#)Num; Per <-- (#)Per;
            Ques <-- (#)Ques.
  NP_f6h
         A: hd <-- #; Num <-- (#)Num.
  NP_h7h
         A: qua <= #.
  NP_h8(send)
         No acts, inits, conds.
  NP_f9(send)
         A: Empty and return Hold
         C: Hold is an NP
+-----------------------------------------------------------+
```

Figure 3-10. Extension of NP network for questions

In transformational grammar, this kind of data has often been handled by "movement transformation" that removes the questioned constituent from its original location in the deep structure and moves it to the front of the sentence. The arcs S_a22a and S_a23a will handle sentences with the form beginning with a question element and exhibiting subject-auxiliary inversion. The hold register in ATN, the distant binding arrows of lexical-functional grammar and the derived categories of phrase structure grammar are other examples of special devices that have been added atop simpler underlying mechanisms in order to handle this dependency.

In ATN grammars, we can think in terms of finding a questioned constituent at the front of the sentence and saving it to be inserted elsewhere when a "hole" is found into which it could fit. This seems like a good candidate for a solution with an initial register settings, like embedded sentences. However, with questions, the passed-down phrase is not limited to a particular role (the subject) but may appear anywhere and any level of depth. Thus the problem can be thought of as that of communication between a higher constituent where it originates and a lower one, where its presence allows a NP or PP to be missing. The communication flow must be bidirectional, since the lower phrases depend on the details of the initial question element, and also on whether a previous constituent has used it.

There are three basic approaches, each with its benefits and drawbacks, to this problem:

Explicit communication. This requires the use of initial register setting mechanism. The element can be passed on down to any constituent that might use it or that could in turn relay it to one of its constituents. The held item can be put into a special role register associated with every type of constituent. In a constituent that uses it, this register is reset to empty. On the arc that called for that constituent, there needs to be an action that checks to see if an item was relayed and not returned, and if so the corresponding register in the calling network must also be emptied.

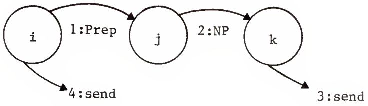
This mechanism can be further modified by the use of a hold register associated with each call to match a network to a word sequence. Since items are taken out of the register at some place in the sentence, this register which is thus temporary is not made a part of the register table that goes in the node structure. This register is special in two other ways:

Automatic emptying. This register is used in a destructive copy fashion whenever its contents are used to set another register(e.g. "Set SUBJECT to HOLD." sets the subject register and empties the hold while check "HOLD is an NP" does not change the content). This mechanism forces a held item to be used only once. The condition at the end on the S_e12 arc which requires hold to be empty ensures that if it is filled (active), it will be used somewhere.

Automatic inheritance. This scheme differs from initialization of registers in that the register is actually shared. When a network is matched(seek arc) its hold register is initially that from the network in which that arc was taken. If the called network uses its constituents then it is empty when parsing returns to the calling network.

Consider Figure 3-9. When there exist an initial wh-phrase(an NP or PP phrase whose head is a QUESTION NP), it is parsed and held in the hold register. The actions associated with the S_a22a and S_a23a arcs are responsible. The remaining part of the job is done by the PP and NP network Figures 3-10 and 3-11 which include both the features enumerated above for distinguishing question forms and the actions required for saving all of the constituents appropriately.

Observe that the NP_f9 and PP_i4 arcs start in their initial states, and since they are send arcs, they successfully terminate parsing of the network without matching any words. They make use of elements from the hold register, and their conditions guarantee that it should be a constituent of the right kind. There is a new kind of action, described as "return . . ." which is used. Rather than go through the normal structure building for send arc, they simply take the indicated node (in the hold register) and return that structure as the result of parsing with the network. These arcs are related to the "VIR arcs" of the original formalism in



Roles: Prep-Object, Preposition

Initializations, actions, conditions:

```
PP_i1j
        A: Prep <-- #.
PP_j2k
        A: Prep-obj <-- #.
PP_k3(send)
        No acts, inits, conds.
PP_i4(send)
        A: Empty and return Hold
        C: Hold = PP
```

Figure 3-11. PP network

that they parse VIRtual element instead of one found at the current position in the sequence of words. They differ in that they appear in the network for the constituent found on the hold list, while the original VIR arcs appear in the network from which the constituent was being sought. They are also related to the "traces" appearing in some versions of transformational grammar, a theory designed to rectify the assumption that the phonological form of a sentence was determined by the surface structure--called into question by phenomena like contraction. It has been found (Chomsky's modification of the extended standard theory) that phonological processes like contraction are controlled not only by the surface form but by something deeper as well. Thus, it has been proposed that both the phonological and semantic components operate not only on the surface structure, but that the surface structure contains "traces" reflecting the relevant information about deep structure and derivation.

Context search and modification. This approach waits until a constituent looks for a constituent and cannot find it, then it searches through the chain above it for the appropriate kind in an appropriate register. If so, it can use it and removes it from the register it was in. This method involves the addition of special operation for context searching and for setting of registers other than the one currently being parsed. It interacts in complex ways with issues of choice and multiple alternatives, since the non-local structure it allows can have unwarranted side effects.

It is also not suited to a system that is purely syntactic, since often it is not obvious that an NP is missing. Many verbs both transitive and intransitive, allow optional second objects, so it takes an analysis of meaning to decide whether there is indeed a NP missing and whether a higher constituent could fill its place.

Global registers. This method avoids the complex search needed to assert if the current constituent is embedded within a constituent that is holding something. It employs a single register that does not belong to any one constituent, but to the sentence as a whole. Any arc that finds an item (possibly nested as with relative clauses), such as the initial NP of a question, puts it into this global register, where an arc corresponding to a missing element can retrieve it. Item(s) in this "hold list" can only be used by constituents below where they were put in. All conditions and actions that involve hold registers operate on this common list, regardless of the network they appear in.

The network of Figure 3-9 cannot deal with questions containing multiple question elements. Newer conditions and arcs are needed to refine the set of sentences accepted.

Relative clauses. This phenomenon is due to the presence of relative clauses appearing as qualifiers following the head of an NP. There are several kinds and we need additional arcs in the NP network to match them. We also need additional states and arcs in the S network to deal with their specialized form. Generally, these clauses can be divided into two groups--those that can be dealt with in a manner similar to embedded clauses and those that require use of a hold registers. In Figure 3-12 an arc NP_h10(s/c)h or NP_h10(s/r)h, for each kind is added as a loop in the final state.

```
+-----------------------------------------------------------+

  Role: Det, hd, des, qua
  Feature Dimensions: as in figure 3-10

+-----------------------------------------------------------+

  Arcs 1-9: see figure 3-10.

  NP_h10(s/c)h
           A: qua <= #.
           I: Mood <-- Rel; Subj <-- COPY based on ↑;
              m-v <-- dummy verb node, word = be.
  NP_h11(s/r)h
           A: qu <= #
           I: Mood <-- wh-relative; Hold <-- COPY based on↑
+-----------------------------------------------------------+
```

Figure 3-12. NP modified for relative clauses

Here the S network is made use of, and an expanded set of MOOD choices (RELATIVE, WH-RELATIVE) is required. Arc NP_h10(s/c)h is for relative clause beginning with verbs in the Present Participle form. This calls for parsing to begin in state S.c with the subject and main verb initialized with a mood of RELATIVE. Since entrance is after the subject-verb agreement test, the exact form of the verb is not important and is not specified.

Consider a relative clause with the verb in present participle form "having height," as in "A box having height." It closely resembles a sentence of the form "NP - V - having height" in which the subject and a first auxiliary are missing, which is a form of the verb "be." By inserting them on the initialization the sequence can be parsed as usual. This method corresponds to almost the transformation called "whiz-deletion" by which the phenomenon is handled in TG. In the process of generation, whiz-deletion removes the subject marker and a following form of the verb "be." Since the operation in ATN is parsing rather than generation we need to insert them. This calls for a new procedure. We recall that earlier, the element used in an initialization was available in one of the registers of the matrix sentence (its subject/direct object). Thus the missing subject resembles the NP in which the S is embedded. We therefore need to construct a node for the NP; for if there existed one its use would result in a circular structure in which the subject of "having height" is the NP--"A box having height." This can be avoided by constructing a node which contains the register contents of the NP being parsed, as they are when that arc is taken. This node is indicated as being a copy so that modules dealing with the meaning of a sentence can connect the structures appropriately (Appendix C-2). This is indicated by the phrase "a copy based on↑" for the value to be put in the register.

```
+-----------------------------------------------------------+
  Roles: Subj, d-O, i-o, m-v, auxs, mods, q-e
  Feature Dimensions: same a figure 3-9
                      Mood includes Relative, Wh-Relative
+-----------------------------------------------------------+
  Initializations, conditions, actions:

  Arcs 1-4: see figure 3-6; arcs 5-12: see figure 3-7;
    arcs 13-18: see figure 3-8; arcs 19-23: see figure 3-9;
  Additional conditions on S_e12(send) (replaces condition
    added in figure 3-9).
      C: If Mood = int or Wh-Relative then Hold = 0
  S_r24a(Rel)
           No init, conds, or acts.
  S_r25b(NP)
           A: Subj <-- #.
+-----------------------------------------------------------+
```

Figure 3-13. Relative clauses in S network

The other group of clauses may be dealt with, with the use of hold registers. Figure 3-13 extends the current S network to handle such cases in which there is a "slot" to be filled in by the copied parent NP. Arc NP_h11(s/r)h is used to match a relative clause and enters the S network in state S.r. This requires the setting of the hold register to a copy of the NP. Thus by use of the same hold mechanism as for questions, we obtain the same degree of generality for embedded use of held items.

If the first element of the S is a relative pronoun it is parsed by the S_r24a arc, and no change is required for the hold register. Later in the parsing, some NP arc seeking the object of "touched" (The plane THAT the box touched) uses the element from the hold register. In the absence of a relative pronoun, the first element of the relative clause is the subject of the embedded S and this is handled by arc

S_r25b. This requires the subject to be entered in the SUBJECT register, with no change to the hold register. The held item can only be the subject of the embedded S only when a relative pronoun is present. We therefore cannot account for missing relative pronouns by having an "r(JUMP)a" arc because this would lead to treating the phrase "the plane the box touched" as containing a relative clause, equivalent to "the plane who(m) the box touched," since the item in the hold register could now be used by the NP parsed by S_a1b arc.

The degree of generality that the hold mechanism offers has been one of the major influences in the design of the ATN formalism. By setting the MOOD to Wh-RELATIVE in a relative clause forbids the traversal of any of the initial questions(S_a21b, S_a22a) arcs. The use of feature register in a manner of a "switch" allows parts of the network to be shared among different kinds of structure while enforcing the integrity of each within its own path.

The extension of the grammar as in Figure 3-13 combines mechanisms for handling embedded sentences of other kinds, questions, passive, and datives. However, many phenomena are not dealt with by the networks. There has been a good deal of work in TG on the kinds of additional constraints needed to eliminate ungrammatical sentences. Generally it is possible to find ATN correlates of transformational constraints, without significantly more(or less) complication.

On Implementation

ATN parsers can make use of grammar that was written without a specific processing regime in mind. The flexibility offered by ATN parsing has made the grammar a basis for models of human language processing. The way that the registers are used forces an orientation to TLR processing. However, current descendants of ATN grammar replace registers with "functional structures" that avoid this directional bias. Even with the ordinary ATN grammar there is still a great deal of latitude in how alternatives are explored. Some ATN parsers work in parallel, others use backtracking, and there are those e.g that designed for speech understanding system(Bates, 1978) that do not work left-to-right, but expand phrases out from "islands" that have already been analyzed(also [MI74]).

The attractiveness offered by ATN has led to experiments like those in Wanner[WA78] which assumes that a person uses ATN-like grammar; and seeks experimental evidence as to the strategy of parsing--serial, parallel, or some complex mixture. The general syntactic Processor [KA73] is a computer implementation incorporating a flexible strategy, which has been used to test detailed psychological hypothesis about human syntactic processing. It utilizes an active chart, which has an agenda of active edges to expand. Rather than to use a fixed order to try items on the agenda, it allows the programmer to include strategy procedures that take other factors into account in deciding what to expand and when.

Structure sharing. A source of inefficiency in a parallel or backtracking parser, no matter how simple the implementation, is that new structures are built for each of the alternatives, even when they have a great deal in common. When there is a choice of arc separate copies of the register table need be kept since modification may be different along the arc. In backtracking, the contents of the table before a choice was made must be preserved. In a parallel mode, the different alternatives must have to work with different tables so that action setting in registers do not interfere. Generally, only one or two registers will differ along the two paths. To avoid such copying of the register table when multiple alternatives are pursued, some data structures allow common parts to be shared. In backtracking, this may be done by using an "association list" instead of a table. This list is a sequence of pairs--consisting of a key and an entry. There can be any number of pairs with the same key. A new entry means adding a new pair to the list regardless of whether there is a pair with the same key. An entry is thus found, by searching backwards for the most recent pair with the key.

The register table can therefore be replaced with such a list. Each time a choice is made all that needs to be saved is the pointer to the end of the list. All actions that take place will affect the list beyond that point only.

If backtracking is necessary, all that is necessary is to restore the previous end position.

For parallel parse a different structure is needed since new additions must be made independently along the paths. A tree structure can be used, in which each node contains a table. Whenever a choice point is encountered, the current node is given a child for each choice. When a register is set the contents are entered in this child node and is not seen by the other alternatives. When a lookup is done, anything not found in a node is looked for in the parent, allowing all the previous settings to be shared.

Feature sets. It is possible to have multiple possibilities in the feature register on the dimension(e.g. NUMBER of "the chicken"), and each possible combination will be the result of a different path through the parse. Here, efficiency may be increased by preserving the nondeterminacy in the structure. If we allow the feature register to contain a set of possibilities we have that, the determiner "the," the noun "chicken" and the NP "the chicken" all have both SINGULAR and PLURAL as their number. In parsing, the subject-verb agreement resolves the appropriate feature. This cuts down on the number of cases in which multiple alternatives are generated, with one or more being eliminated by later checks.

This however calls for the modification of the formalism for setting and the use of feature registers to allow sets of choices, and the specific conditions and actions in the grammar must be modified accordingly.

Arc ordering. By careful choices of ordering the grammar can "shape" the direction of the parser. In systems that explore alternatives serially, it is necessary to choose some order in which to try the alternatives. In some parsers, this has been used to increase efficiency by ordering in such a manner that those more likely to lead to a successful parsing is tried before those of rarely occurring structures. This can increase the speed with which common structures are parsed. Since this is combined with some sort of backtracking, it will not prevent the other alternatives if they are needed.

Weights can be used to assign a total value to a path. The parser uses the information to score alternative paths, with the effort going towards the path with the highest value of combined weight. Factors of meaning can also be used to determine the value of a path. Tests may also be factored (as in the BBN parser of Bates, 1978); wherein each arc test is split into two--one was context free(to test only features of the current word) and the other was context-sensitive(involving register context).

A simpler method has been the grouping of arcs into sets--a set of mutually exclusive arcs can be grouped--such that if any one member of the set succeeds the others will necessarily fail, and they need not be tried even if the parser should back up. This can cut search space with no loss of generality. The Burton grammar-compiler (1976) allows a subset of the arcs from any state to be grouped.

Explicit structure manipulation. In network that includes right-recursive structure there is potential for ambiguity. In some grammars strings of PPs can lead to many alternative parsings. In the sentence:

"I saw a man on the hill with a telescope"

Analysis of each noun group finds, the man has a telescope; in others, I used a telescope; and in others the telescope sits on a hill. In some, I am on the hill, while in others, the man is. A phrase with one more level of preposition groups has 14 possible parsings.

In some ATN grammars, such strings can be handled by putting conditions on the NP_h7h and S_e9e arcs to prevent them from being taken under certain conditions (as when the NP or S they are in is already the rightmost descendant of a PP). Action associated with the send arc on S would then call other modules, including those for meaning to decide the distribution of modifiers among embedded phrases, possibly leading to more than one parsing.

In discussing possessives (appendix C), we add the possibility of having actions that directly manipulate the underlying data structures of the implementation. This is commonly done since it provides a "cheap" way of dealing with non-standard structures. In addition to those uses (possessives, conjunction, and the building of result structures), structure manipulation has also been used in techniques for ambiguity reduction.

CHAPTER 4

DESIGN CONSIDERATIONS AND AN ATN BASED SYSTEM

Computerized Implementations

We have discussed concepts related to syntax and parsing in a conceptual manner. We now discuss some of the existing systems which can be divided into several basic groups. These systems cover a time period starting with the late 1950's.

Machine translation. The first computational work on natural language was done in pursuit of machine translation, an effort that was heavily supported in the 1950's and early 1960's. After a period of little activity, machine translation has re-emerged. Commercial systems, such as SYSTRAN [TO77], are being developed, and there are current efforts to specify a major translation system to be built under the sponsorship of the European Economic Community [WI83].

Integrated question-answering systems. From the mid 1960's to the mid 1970's, much of the new work was centered around the development of comprehensive systems that integrated syntax, semantics and reasoning to carry out some kind of question-answering or cooperative task. The focus in these systems, was on the broad coverage of linguistic phenomena, including complex syntax. Toward the end of the period, there was a concerted project to develop a speech understanding system, which again focussed on the integration of multiple knowledge sources. Some question-answering systems have been integrated into systems for computer aided instruction, such as SOPHIE [BU79].

Data base retrieval. In the last few years, there have been two different kinds of work in natural language on computers. One trend has been towards pursuing of deeper problems of meaning and conversation, concentrating on issues other than syntax. The other trend has been towards system that have a limited capacity to handle complex syntactic deduction, but can be used as a practical front end for data-retrieval system. These systems have generally made use of well-understood techniques to cover a subset of natural language that is incomplete but "habitable" in that a person using the system will quickly learn what kinds of things are handled and what to avoid.

Many natural language data base interfaces (e.g. LIFER [HE77], and REQUEST [PL76]), are designed to to operate as front ends for commercially available data base systems, while others (e.g. the much earlier REL [TH69] system) were developed with their own data base formalism.

Text analysis. There have been a number of applications that are not interactive, but involve the syntactic processing of bodies of text. One purpose is to provide a basis of information retrieval, in which the analysis can be used either to retrieve pieces of text or to extract a structured data base directly from textual information such as medical record. The Linguistic String Parser[Sa81] was developed principally for such applications. There are systems designed to aid in writing(e.g. EPISTLE[MI81] and the Writer's Workbench [CH81]), which use syntactic analysis techniques to find possible errors in text being prepared. They do only a limited form of parsing, but are able to detect certain classes of errors, such as improper subject-verb agreement and pronoun cases.

Text generation. There has been some work on the generation of text by computers, including the output of question-answering (Simmon and Slocum, 1972). Some of these are based on transformational grammar (as in Bates and Ingria, 1981) that generates sentences in a tutorial program for teaching English to hearing-impaired children. The difficult problems in generation are those of meaning and context rather than those of syntax. Usually, a program for generation can get by with a repertoire of output structures that is much limited than full English, handled by a simple finite state or phrase structure generator.

Theory-motivated systems. In the history of computational linguistics, there have been projects dealing with parsing as a theoretical problem, not associated with a particular application. One such project is the Harvard Syntactic Analyzer[KU62], which was an early version of top-down backtracking context-free parser. A grammar of over 3000 rules was developed for English. The General Syntactic Processor[KA73], definite clause grammars, and MCHART[TH81] are examples of recent attempts to define a parser for a particular class of grammars.

The motivation in developing computer systems has always been dual. On one hand, they have practical uses. On the other, they can be viewed as hypotheses about the nature of the human language processing, and their mechanisms can be the basis for psycholinguistic theory. Systems motivated by theoretical considerations do not emphasize wide coverage of practicality, rather they attempt to capture in their design the same kinds of generalizations that linguists and psycholinguists postulate as theories of language structure and use.

Language learning. There have been some programs that attempt to learn the grammar of a language through examples. In general, these have been based on assuming that the underlying grammar is very simple (finite state or C-F with only a few rules), and have made no attempts to deal with the complexities of real language. Many systems, including the one described in this dissertation, have allowed the user to enter new vocabulary items and specify their meaning and syntactic class. Some infer the class of new words by assuming it is in the class that would make a sentence grammatical. Others, use an English-like dialogue to allow someone to define new words and syntactic properties (we also allow this)--including case frames associated with verbs--without needing to know the linguistic formalism underlying the program.

Issues

In designing a system for analyzing natural language inputs, there are a number of issues to be addressed. There are tradeoffs of power versus efficiency and simplicity, decisions as to what constructs need to be handled, and different aspiration levels for the generality of the system outside of its specific intended use. These decisions are interrelated, each having consequences in a number of areas.

Completeness

Over the years there have been systems whose parsing is not based on a systematic approach to grammar and syntax. Instead, they contain a collection of algorithms and strategies that perform a different analysis for the purpose of the overall system.

There are those who argue that a full syntactic analysis is not necessary in practical systems, since semantic considerations can lead to a full analysis in cases where the syntactic structure is not too complex. The general attitude is that although practical analysis can be useful--and sufficient for some specified tasks--any fully general language system will need a syntactic parser that uses a large grammar based on systematic formalism.

In practical analysis, there is a tradeoff between specificity (assigning structures only when they are appropriate) and exhaustiveness (assigning structures to as many inputs as possible). Any system with low specificity will often assign structures that are not appropriate to an input. Oftentimes, the assigned structures will be plausible but the information that was discarded would have pointed to a different structure intended by the operator. Systems doing partial analysis are most useful in applications where high specificity is not required (e.g. document retrieval). A good example is the PARRY[CO74] program for psychiatry interview simulation. All the system needs is only to be able to do something for every input.

Basic Framework. We now seek to establish an overall architecture of the system and grammar after we dispense with a few other design issues.

Augmented phrase structure grammar. One common approach is to augment C-F grammar in much the same manner as ATNs augment recursive transition networks. The augmentations associate features and/or registers with nodes and specify conditions and/or actions associated with rules.

Transformational grammar. Attempts have been made to use transformational grammar directly in parsing a system. The major advantage is the existence of a large body of work within the transformational framework that could be used to build systems covering a very wide syntactic phenomena.

Pattern matching. Some systems are organized around the simple ideas of pattern matching, along with additional ad hoc processing techniques. Generally, these are systems that do only partial syntactic analysis, in conjunction with some kind of semantic analysis.

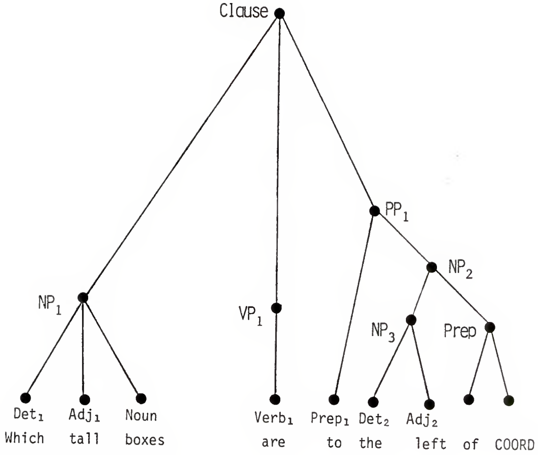
Situation-action rules. In standard parsing, there is some kind of grammar describing the kind of structure to be found and a parsing process that uses the grammar. It is however possible to organize the system around a set of procedural situation-action rules instead, each triggered independently by features of the situation and can act to change the situation. Such systems are less structured than standard grammar, and often are used for partial analysis, or in cases where semantic considerations drive the analysis process.

Form of assigned structures

In building a parsing system, one usually has in mind some purpose for the parsing. Some other components of the system will make use of the analysis to answer questions, find documents, store, generate sounds, produce a translation, or as in the present case, drive a graphics display system. There is a good deal of latitude in the division of labor among components. The syntactic analysis may be superficial, leaving more work to the semantic analyzer, or they may provide a deeper analysis that is more immediately usable in the activities of the system. It may try to achieve completeness, trying to preserve all of the information about the sentence, or can be boiled down, leaving only that which is known to be relevant for further analysis. Types of strategies include:

Annotated surface structure. The most straightforward result of parsing is a tree structure corresponding to the parse. In a simple C-F system, the result is a phrase structure tree. In an augmented system, the tree can be annotated with features, as in the PROGRAMMAR system used in SHRDLU[WI72] and the system GATORGFX described here, in which each node has a list of features. The subject "Which tall boxes," and prepositional agent "to the left of (5,3)" are left in their surface positions in the tree--the syntactic parser does not move them around, but simply adds the feature PASSIVE to the node containing them. The semantic analyzer observes this feature and uses the correct elements for the deep subject and object of the verb, as shown in Figure 4-1.

Deep syntactic structure. To simplify the semantic analyzer, many systems do some kind of transformation on the surface tree to produce a syntactic structure--where the phenomena like passive voice and implicit subjects have been accounted for. In systems like LUNAR[WO73], the generated structure is an ordinary phrase structure like the deep structures of transformational grammar. In others, the structure is registers, like the role and feature registers that were described. A particular kind of structure generated by some parsers is based on CASE frames(to be described later). Thus rather than return a clause structure containing subjects, objects, and prepositional modifiers, these systems combine them when possible into a case structure based on the particular verb.



```
Clause(which boxes are to the left of 5,3 ?)
 Major Question Wh-Transitive Passive Subject-Question
                Agent

    NP1(which tall boxes) NP Determined Indefinite
         Question Plural Subject
       Det1(which) Determiner Question-Determiner Singular
                    Plural
       Adj1(tall) Adjective
       Noun1(boxes) Noun Plural
    VP1(are) VP Verb Auxiliary Intensive Be Plural Present
    PP1(to the left of 5,3) PP to Agent
       Prep1(to) Preposition To
     NP2(the left of 5,3) NP Definite Singular Prep-obj
       Det2(the) Determiner Definite Singular Plural
       Adj2(left) Adj
       Prep(of) Preposition Of
      NP3(5,3) NP Definite Singular
       Noun (5,3)
```

Figure 4-1. Surface structure produced by GATORGFX

Systematic non-syntactic structure. Some systems do not have a need to produce a complete syntactic analysis. Each constituent can be analyzed for its "content" as it is parsed, and that content used to build up an overall semantic analysis. In such systems, the structure is based on a semantic theory and each syntactic constituent simply contributes its part to the evolving semantic structure.

Arbitrary structure building. Some systems do not attempt to provide any systematic resulting structure. This allows the grammar writer to associate a structure building procedure with each rule, using the underlying implementation language(e.g. LISP). A particular writer could choose to write actions that produce annotated surface structures, deep structures, semantic structures, or anything else. These result in simpler systems and have a greater generality while placing a correspondingly higher design burden on the user.

Ambiguity

In the assignment of meaning to sentences, complication may arise where words are polysemous and hence result in in multiple successful parsing (ambiguity). Determination of the correct meaning may require information implied by the general subject area or explicitly stated in the previous sentences of the discourse. In dealing with this problem, there are three choices: Ignore it-- insist that the grammar give only one parse for any ungrammatical sentence; use the first parsing found--maybe providing a clue to the parser as to the order in which to try things; and find multiple possibilities to choose among. In case of the third, the simplest procedure, is to respond to all of the interpretations, or to present the alternatives to the user and let him indicate his intention. Alternately, we could generate a response for each interpretation and eliminate those for which there is no sensible response.

The simplest kind of word ambiguity is syntactic ambiguity which results when a given word has two or more grammatical parts of speech as in:

Draw a deep rough box ..... (1)
Draw a deep blue box ..... (2)

Here, ambiguity is due to the fact that "deep" may be either an adjective or an adverb modifying box or rough respectively. In (1) we resolve this ambiguity by noting that the adverb "deep" may only modify color (semantic restrictions). Thus, since in the sentence "rough" is not a color, "deep" must be an adjective. The ambiguity due (2) is more difficult to resolve, however, since blue is a color. In GATORGFX, we pick consistently one interpretation and use the backup ability of the program to make corrections in the next sentence if the choice was incorrect. As mentioned above, by maintaining this consistency, the user soon learns the interpretation which the system will choose and thus learn to get the desired response.

Another type of ambiguity is semantic ambiguity, resulting when a word has two or more meanings, all of which are the same part of speech. This may be resolved as discussed above or by examining the context in which the ambiguity occurred.

This however, is rarely a problem in our "world" as the context of all words is known a priori by the nature of the world itself.

The other issue is that of exploration of the space of alternatives in searching for a successful parse. This issue is more complex and approaches for dealing with it include:

Parallel parsing. It is relatively easy to write a parallel system if efficiency is not of prime consideration. Systems based on augmented phrase structure rules make use of parallel parsing techniques. However, with grammars more complex than C-F grammars, it can be quite difficult to provide for sharing of common elements that make parallel implementations relatively efficient (as we discussed in Chapter 3).

Chronological backtracking. This is one of the simplest strategies to implement. Parsers making use of more complex grammar formalisms (e.g. ATNs) have often taken advantage of this simplicity to keep the overall system from going beyond reasonable complexity bounds. The natural recursive structure of some languages can be used to implement the backtracking involved in keeping track of register assignments. Some (non parallel) parsers use arc ordering to lead the parser into the exploration of more likely alternatives first (again as we previously discussed).

Explicit backtracking. Some systems, do not have uniform means of keeping track of choices, but instead include explicit actions in the grammar to go back and try something else. Thus as an example in our system, a sentence like "What color is the box to the left of the circle?" "color" can be treated as a verb when in fact it is a noun. When the main verb is recognized, the system explicitly goes back and removes the verb from the group and treats it as a noun. This strategy runs more efficiently than uniform backtracking, since it does not keep choice records at every alternative. Its rather difficult to work with, since we must anticipate those places where wrong paths can be taken and must carefully undo all the effects of following them before going back. It does not distinguish knowledge about the structure of grammatical sentences from the process of parsing as clearly as does the ordinary ATN formalism.

Deterministic parsing. There are specialized forms of grammar that can be produced for every language. A useful property of some formal languages is that there exist C-F grammars for them that can be used deterministically. In a LTD parse, it is always possible to decide which rule to use in expanding a symbol by looking ahead in the input string some fixed number of words. There are choices of rules on these "bounded-context grammars," and the choice can depend on varying amounts of lookahead and recall of what has been done so far. Natural languages do not fall into this class, but there are parsers which treat natural language in a bounded-context style, assuming that at every choice point it is possible to choose the right alternative deterministically.

The choice of strategy for exploring multiple alternatives is often guided by the way the syntactic parser will be integrated with other components of the system. If semantic analysis is done on a phrase-by-phrase basis as the parsing progresses, then a backtracking strategy will often work well, since the semantic analysis eliminates false paths early. But, if all the parsing is done before any further analysis,some kind of parallel strategy is called for, since all of the alternatives must be available.

Syntactic coverage

No system (or linguistic analysis) covers all of the phenomena of natural language. The designer needs to decide which things are critical and which can be ignored, and this decision is guided both by theoretical concerns and by practical experience.

In deciding on the breadth of coverage to be attempted, there are a few issues to be considered:

Grammaticality checking. In the formal grammar of Chapter 3, there was an emphasis on having the grammar precisely characterize the sentences of the language. Any string of words that was not a legal sentence would be rejected by the recognizer or parser. In practical systems, this ability is not vital. As a matter of fact, a system that accepted ill-formed sequence might even be better provided it were accurate in assigning the right corresponding form.

There are strong arguments for and against precise grammars. Those who argue against them draw analogies to the human capacity for understanding a wide range of "ungrammatical" utterances. Those for, point out the vital function that a grammar serves: that for every rule, no matter how obscure, there exist sentences that would be ambiguous if the rule were not taken into account, but whose structure is not in fact ambiguous.

In speech understanding system, grammaticality checking can even be more important, since there exist uncertainty about the very words of which the sentence is formed.

Complex clause embedding. Much of the work in transformational grammar, as we shall see, has been motivated by the analysis of complex embedded sentences. In the design of practical systems, one has the choice of simply rejecting such sentences, in place of a series of simpler sentences. Systems built from a more theoretical perspective (using TG or ATNs) tend to include these complexities, while those for practical applications--often using augmented phrase structure grammar--do not.

Phrases and idioms. Much of our everyday language is made up of stock phrases that are not understood by a direct analysis of their structure. Some such as "teeny bit," could be syntactically analyzed but are used so frequently that they act like atomic vocabulary items. In their evolution, over passage of time, they develop their own unique meaning. In theoretically oriented systems, these have tended to be ignored. In practical systems, a moderate stock of phrasal items can account for a great deal of complexity encountered in the inputs given by naive users. In our system, word idioms are treated as morphological units rather than formalizing rules to allow their aggregate meaning to be discerned from meaning of constituents. While initially, the meaning may be reflected in the constituent words, passage of time, and change in usage may cause the the idiom to diverge from its initial "assigned" value. Consequently, often used multiple-word idioms become contracted through normal speech slurring into single word entities which may be used again in the evolutionary cycle. From a pragmatic point of view, it is much easier to parse at this idiomatic level than at a single-word level because the associated grammar is much simplified.

Specificity of domain

In designing a system for a specific application, one has the possibility of taking advantage of peculiarities of that application or its subject domain in order to make the system work better. The earliest system such as STUDENT[BO67] was designed specifically for a single problem area. STUDENT analyzed every sentence as a linear algebraic equation, since its task was to solve algebra word problems. Later systems are more general in their treatment of syntax while remaining domain-specific in their treatment of meaning and the overall organization of their interaction with a user. However, there are a number of ways in which specific knowledge of the subject area can aid in building an effective parser.

Special patterns. In special areas there will be phrase forms and idioms that need to be treated specially. Rather than adding complexity to the grammar for the ordinary constituents, it may be more straightforward to introduce new categories such as "COORDS" in our system. Similarly, the purpose of the system (e.g. to answer questions from a data base) can lead to the common use of standard phrases like "What color is Figure 1?" Special treatment of these can serve to handle a large number of inputs that would otherwise require a more complex grammar.

Semantic lexical categories. In addition to having syntactic categories that represent special classes of phrases, it is possible to organize the dictionary so that words are classified according to semantic as well as syntactic criteria.

By writing a grammar that is specific to a particular topic domain, we can let the syntactic component do more of the work, producing fewer proposed parsing for semantic analysis. With some system organization strategies, this reduces the problem of ambiguity and makes the overall system more efficient. The importance of introducing semantic categories into the grammar is related to the use of a system organization in which parsing and semantic analysis is done in separate stages. A system that does semantic well-formedness checks on constituents as they are being built up can have the same effects on parsing while keeping syntactic categorization distinct from semantic categorization.

System Engineering

This issue lies outside of the theoretical domain of language and grammar, but is important in the building a usable system. In comparing the relative merits of different system, practical considerations of the use of the system prevails over theoretical issues. Two different classes of users need be considered: the ultimate end users who will interact with the system and the specific application designers who will build grammars and enter knowledge for topic domains. A one-user system could to some extent ignore the latter class.

User interface. For the end-user, an ability to go back and edit a request and the presence of spelling corrector can be extremely convenient. Even a system with marvelous grammar will not be convivial if it demands that a user type in a full line with no typographical errors.

Efficiency. There are theoretical efficiency of different parsing strategies. The theory of computation deals with wholesale order of efficiency--be it O(linear), O(logarithmic), O(polynomial), or O(exponential). This is usually measured with respect to the length of some particular input. In this analysis, constant factors are not usually considered. Two different implementations may be equivalent in the sense that they both take a time proportional to the square of the length of the input, but one may be orders of magnitude faster.

Speed is often gained at the cost of generality, and there are many external contributing factors, such as the particular machine used in processing and the amount of effort devoted to optimization. In systems designed for interactive use, the important constraint is that the response time be acceptable. In most cases, the syntactic analysis constitutes only a fraction of the overall response time, so optimization is not as important on the whole. In applications such as processing for text retrieval, the speed of the parser is more critical and the systems are correspondingly more tuned to efficiency considerations.

An ATN System

In the previous chapter we presented the ATN formalism. With augmentations--role and feature registers, conditions and actions--we were able to context-modify the topology of the recursive transition networks and thus ATNs have the generative power of unrestrictive grammars. The send arc for each constituent may have and associated "tree-building action" which takes selected register contents and substitutes them into tree patterns. In Figure 4-2 we illustrate a simplified ATN grammar which is able to describe a wide class of English sentences. Since language understanding involves inverse transformation from surface to deep structure this grammar has been so structured. The branches between the nodes of the network are conditioned by both the syntactic part-of-speech and the context registers. Parsing a sentence involves finding a complete path through the sentence network with appropriate calls to sub-network representing non-terminal symbols.

Semantics

Using the grammar, the sentence:

The	elephant	ate	the	bananas
[+def]	[+animal]	[+past]	[+def]	[+fruit]
	[+tall]	[+trans]		[+yellow]
	[+heavy]			[+long]
	.			.
	.			.
	.			.
	.			[+creamy]
	[+Africa]			[+sweet]

has the hierarchical structure representation shown in Figure A-7. We may then merge the features shown to obtain a deep structure of Figure 4-3 representing the meaning which the system ultimately associates with these words. With this kind of semantic knowledge, the program will reject or

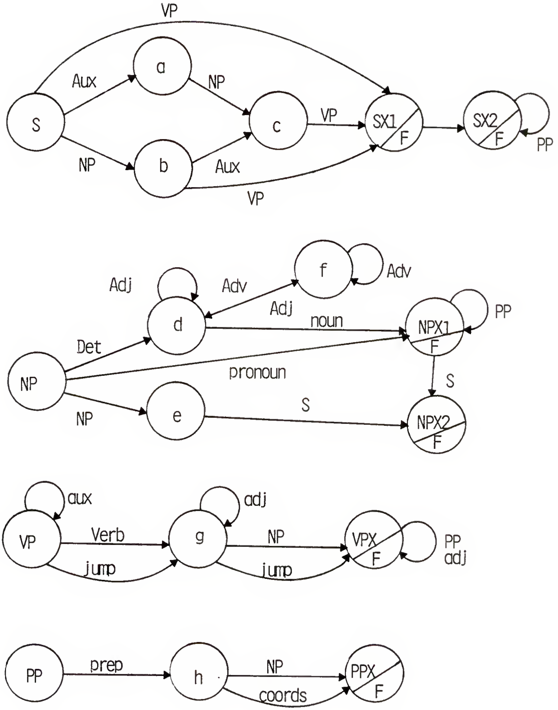


Figure 4-2. Augmented transition networks.

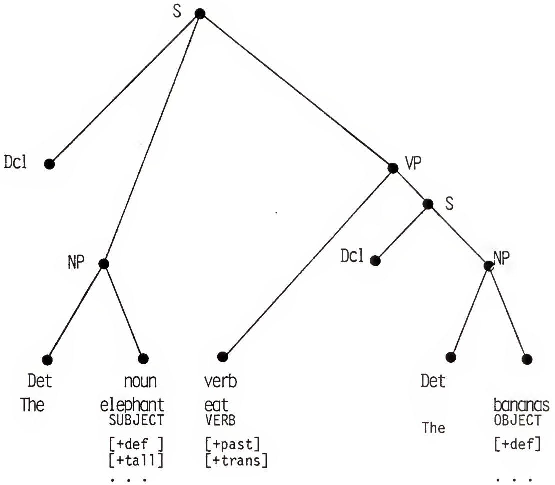


Figure 4-3. Deep structures produced.

accept inputs based upon knowledge of its meaning even though it is syntactically correct to the parser. For instance, if the roles of the subject and object were reversed in the above sentence, the semantic analysis will cause a rejection of the sentence.

Programs of this nature, is exemplified by SHRDLU[WI72] which typically combines the syntactic and semantic analysis into one process. This is done by allowing previously parsed words to define extra tests (different from syntactic) on the branches of the network yet to be traversed. The PROGRAMMAR system used in SHRDLU was strongly influenced by systemic grammar and therefore makes extensive use of feature markings, such as those above on both words and constituents, organizing them into explicit choice systems. The overall SHRDLU system emphasizes integration of syntactic, semantic, and reasoning processes, and this is facilitated by the fact that conditions and actions are arbitrary LISP procedures that can deal with information of any type, based on either syntactic features or semantic analysis. Meaning checks--involving the current state of the parser's world as well as its general knowledge--are interspersed into the parsing process.

As examples of the semantic restriction in action, let us consider the phrase "An elephant is eating." The parsing process recognizes a singular determiner "an" and requires that the noun ultimately parsed have a singular form. Once this agreement is established, the noun phrase, "An elephant," is established to represent a singular subject. This condition necessitates a singular form "is eating" since "are eating" would be incorrect.

Associated with the verb "eat" is the feature that its subject is [+animal] and the object is [+edible]. While banana may not have this feature explicitly, it could be inferred from the feature [+fruit] by an implication mechanism within the program. Such mechanism reduce the amount of explicit information which must be stored about a word although they require more execution time to operate (as we alluded to earlier).

An extension of the deep structure is that of the Case. English relies primarily on the order of the constituents in a clause to indicate their roles. In many languages, explicit markings such as particles and the morphological structure of words are used to represent the relationship between participants and an action or state.

There is a choice system of six cases called nominative, accusative, genitive, dative, instrumental, and prepositional. There is a complex set of rules for assigning endings to nouns in different cases, and this is a function of the particular noun, its number and gender. There is another set of rules describing how the different cases are used. The syntactic cases in a language have a certain degree of correspondence with meanings, but its far from being a one-to-one relationship. The same case is used for a variety of meanings, and often things that seem similar are expressed with different cases. Languages have different number of cases and differ greatly in how they divide up the choices of meaning to be conveyed. It is thus important to recognize that "surface case" as they are called, does not directly reflect conceptual meaning, but rather is a mechanism used to connect conceptual relations to surface structure.

Each verb in the language has one or more ways of setting up a correspondence between the things in the conceptual scenario it describes and the underlying case frames provided by the grammar. There have been attempts to find semantic principles that can explain differences in how verbs accept cases, but there seems to be a great deal of irreducible idiosyncracy.

Besides the simple subject and object relationship discussed above, Simmons has proposed the "deep case structure" of verbs. By recognizing that the allowed structure of a sentence is primarily influenced by its verb (e.g. transitive verbs require an object to act upon), he defined a number of cases to relate the verb to the noun phrase. The case structure also contains information about its "modality" which is further broken down to represent Tense, Aspect, Mood, Essence, Manner, Modal, and Time features of the verb [SI73]. There were five cases proposed: CAUSAL-ACTANT(which includes both the AGENT--instigator of an event and INSTRUMENT--stimulus or immediate physical cause of an event), THEME (used to cover a variety of participants including OBJECT, which is an entity that moves or changes or whose position or existence is in consideration), LOCUS(serving as LOCATIVE--location or orientation), SOURCE (place from which something moves), and GOAL (place to which it moves). In his system, the same role can be filled more than once. Since each verb may have several cases, existing within a sentence, these restrictions may be used up to divide up the different roles to determine which noun group belongs in which case; and whether sufficient information exists within the sentence to enable comprehension or must be implied from the discourse. These restrictions are easily incorporated in our conceptual model as will be shown later.

The important feature of case frames from our point of view is the fact that each verb contains restriction about the types of noun phrases which is allowed in each case relation to the verb.

Knowledge Representation and Semantic Networks

Critical the semantic analysis of a sentence is the formalism used to represent knowledge. Winograd used word features and stored knowledge about his "block world." A person was allowed to give commands, state facts, and ask questions concerning a series of actions being displayed. Declarative sentences added new assertions to the data base.

Since AI research methodology involves the design of programs that exhibit intelligent behavior, researchers have often taken a pragmatic approach to the subject of knowledge, focusing on improving the behavior of their programs. The "representation of knowledge" is a combination of interpretive procedures and data structuring which when used in the right way in a program, leads to "knowledgeable" behavior.

Knowledge

The kinds of knowledge that may need to be represented in systems may be (1) objects: since these may represent facts about our world. There should be ways to represent objects, their classes and categories, and description, (2) actions and events in our world, with representation formalism which may indicate time course of a sequence of events and cause-effect relations, (3) knowledge about what we know (meta knowledge).

The goals of systems can be described in terms of cognitive tasks like recognition of objects, answering questions, and manipulation of intelligent devices. The use of knowledge in programs may require (1) acquisition of more knowledge--new data structure need to be classified before it is added to the data base so it can be retrieved later when it is relevant. To improve its knowledgeable behavior integrative process must take place during acquisition, so that new structures can interact with the old, in some possibly inferential way, (2) retrieval--this involves the determination of what knowledge is relevant to a given problem. This may be crucial when too much knowledge is ascribed to the system. Ideas that have been developed along this line, might be termed "linking" and "lumping" i.e. if it known that one data structure is going to entail another in expected reasoning task, an explicit link is put between the two; when several structures are going to be used together, they are lumped into a larger structure, (3) reasoning--when a system has not been told explicitly how to do a certain task it must reason from its repertoire of knowledge. The kinds of possible reasoning in a representation are (i) formal--which involves the syntactic manipulation of data structures to deduce new ones following prespecified rules of inference (ii) procedural--uses simulation to answer questions (iii) analogical--the natural mode of thought for humans, but rather difficult to implement in AI programs. (iv) meta-level--uses the "knowledge about what is known," i.e. the extent of ones knowledge about the importance of certain facts.

These are interrelated, for when new knowledge is acquired, the system has to be concerned with how that knowledge will be retrieved and later used in reasoning.

There are different representation formalisms and as with any, the choice of the primitive attributes of the domain that are used to build up facts in the data base strongly affects the expressive power of the scheme. Most research assume that what needs to be represented is known a priori and the job then is to figure out how to encode the information in the system's data structures and procedures. These schemes include (1) state-space search--used for problem-solving and game-playing programs. The search space is not a representation of knowledge but a structure of a problem in terms of the available alternatives at each possible state of the problem, (2) logic: the rules of inference is the process of making deductions from facts; and the most important feature is that deductions are guaranteed correct to an extent that other representation schemes have not yet reached. The popularity of this scheme in AI research is that derivation of new facts from old can be mechanized. However, when the number of facts become large, there is a combinatorial explosion in the possibilities of which rules to apply and to which facts at each step, (3) in a Procedural representation, knowledge about the world is contained in procedures--programs that know how to proceed in well-specified situations. For instance, in parsers for natural language system, the knowledge that a NP may contain articles, adjectives, etc., is represented by calls to routine that know how to process them.

All AI systems use this representation at some level of their operation, and general consensus gives a legitimate role for this scheme [BA81], (4) production systems--the idea here is that the data base consist of rules, called "productions" in the form of condition-action pairs. (5) the most recently developed scheme is the "frame." This is a data structure that includes declarative and procedural information in predefined internal relations. A likely frame is selected to aid in the process of understanding a current situation be it dialogue, scene, problem, and the frame tries to match itself to the data it discovers. If it is not applicable, it could transfer control to a more appropriate frame. (6) the semantic net was developed by Quillian [QU69] and others for storage of information. It was invented as an explicitly psychological model of human associative memory.

Researchers in computer science have been more concerned with developing functional representations for the variety of types of knowledge needed in their systems. Because of the diversity in these goals, there is no simple set of unifying principles to apply across all semantic net systems. Information in the nets is stored in much the same way as in a dictionary, with words defined by references to other words which are similarly defined. Thus, a person who lacks the understanding of English, could wander through the dictionary in pursuit of accurate definition of a word unless he had a a dictionary which could provide an absolute meaning--translation from his language--for the word. This also is a drawback in semantic nets--since it lacks any absolute reference for meaning--the meaning of a word is defined in its relationship with other words. Nevertheless, we will show later how the beauty of semantic nets as models of word meaning fits our multi-dimensional concept space model.

Semantic Networks

A net consists of nodes representing objects, concepts, situations and events in the domain, and links between nodes, representing their interrelations. Node-and-link structure captures something essential about symbols and pointers in symbolic computation and about association in the psychology of memory. A key feature of this representation is that associations can be made explicitly and succinctly--relevant facts about an object or concept can be inferred from nodes to which they are directly linked, without search through a large data base. This ability to point directly to relevant facts is salient with respect to ISA, ISPART and SUBSET links, which establish a property inheritance hierarchy in the net. The relation ISA, ISPART, are partial orderings on their domain. One of the most important relationship is transitivity i.e.

$$\forall\ a,b,c\ ISA(a,b) \wedge ISA(b,c) \Longrightarrow ISA(a,c)$$

There are no convention about the meaning reflected in the nets. The interpretation of net structures depend solely on the program which manipulates them.

Suppose we wish to represent a simple fact "A cube is a box."

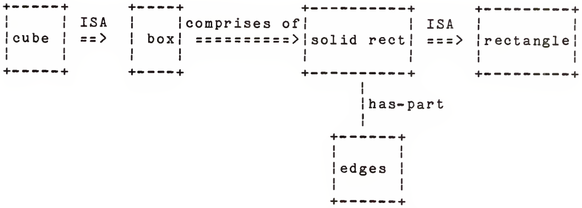


Figure 4-4. An example of semantic network

From above we can deduce that cubes have edges. The ease with which it is possible to make deductions about inheritance hierarchies is one reason for the popularity of the network as a knowledge representation. All that is necessary is to trace up the ISA-hierarchy, assuming any facts asserted about higher nodes on the hierarchy can be considered assertions about lower ones also, without having to represent these assertions explicitly in the net. This kind of reasoning is referred to as "property inheritance," and the ISA link is often referred to as "property inheritance link."

In a semantic network, all of the features possessed by a given noun are indicated by symbolic links to the definition of words representing those features. The links by their nature, can encode only binary information, and this precludes a quantitative representation of "grey meaning," however, and unless all such feature words expressing the same dimensionality are ranked by their varying degrees of expression of this dimensionality, a semantic network cannot express absolute meaning.

Nodes may also represent situation and actions as well as objects and sets of objects [SI72]. Each situation node can have a set of outgoing arcs, called "case frames" which specifies the various arguments to a situation predicate. Many semantic network systems use sets of case arcs motivated by linguistic considerations, for example, general case structures for agent and object.

Our concept-space representation for OBJECTS could be considered as a type of semantic network in which all OBJECTS are connected to tokens expressing the individual features of the space by weighted links. These weightings thus allow a continuous specification of the degree to which each OBJECT embodied each feature.

This concept-space representation may also be used as a mechanism for resolving Case Relational dependencies in Case Structure analysis. The verb plays a central role in determining the structure of a sentence. Each verb can specify one or more case frames, restrictions on what surface structure can be used and prepositions that can be used for specific roles. Prepositional phrases appear as modifiers of sentences and noun phrases and carry much of the information about what happened and who the participants were. In order to make sense of a sentence, the grammar must be able to assign each modifier to a role, often in situations where simple syntactic rules allow a great deal of ambiguity. By having case frames included in the dictionary entries for verbs, the parser has an additional source of knowledge to guide the analysis of the input sentence into a constituent structure.

Verbs that are closely related in meaning can differ arbitrarily in how they appear in clauses. Since each verb in this model may permit only certain types of "things" to stand in each Case Relation to it, this comparison, with Relational dependencies, may be made on the basis of an OBJECT's concept-space location. To each Case Relation of each verb, we associate a disjunction of hypersolids in the concept space which represented the allowed types of objects for that case relation. Wherein a definition point of a given object is included within one of these hypersolids, is sufficient to allow a given Relation, and the ability to quantify this test makes it adaptable to computer implementation.

Conceptual Dependency and Concepts

A great deal of information conveyed about a sentence may be found only within its context with other sentences of the discourse, or by intuitive inferences based on knowledge of the subject area; since English sentences sometimes rarely provide all of the information necessary for a total comprehension. Thus, a level of understanding higher than the syntactic and semantic understanding is that of conceptual dependency (relationship)(CD) within the sentences of the discourse. Shank [SH73] has proposed a model of conceptual understanding to account for these inferential abilities. Its most important distinctive feature, is the attempt to provide a representation of all actions using a small number of primitives. He contends that it is the business of an adequate representation of natural language utterances to capture their underlying conceptual structures. The requirements are that (1) the representation be unambiguous, even though the input may contain syntactic ambiguity and (2) the representation be unique i.e. distinct sentences with the same conceptual content should have the same representation. This has been characterized as the basic axiom of the system and has been identified as accounting for human abilities to translate and paraphrase text.

His representation for meaning (CD) is based on language-independent conceptual relationship between objects and actions. His set of conceptual roles bears a strong resemblance to the more syntactically motivated notions of case. The fundamental difference being that roles are filled by conceptual entities (objects, and acts) rather than syntactic ones (NPs and clauses). Some roles require an act as their filler. His proposal for conceptual structures relies principally on a set of 11 primitive action verbs (ACTS) which include physical and mental acts, and those that are characterized by by resulting changes in state or that are used as instruments for others. Conceptual dependency generally expresses the meaning of an action verb only in terms of its physical realization. Besides these primitive acts, there are other categories or concept types.

Concepts are either nominal, an action or a modifier. Nominals are thought of by themselves without relating them to some other concept. Thus a word that is the realization of a nominal tends to evoke a picture of that real world item in the mind of the respondent. The nominal concept is thus referred to as a Picture producer (PP). An action is what a nominal can be said to be doing, while a modifier makes no sense without the nominal or action to which it relates. It is a descriptor of the nominal or action to which it relates and serves to specify an attribute of the nominal or action. Modifiers of action are AA's (Action Aiders), which are attributes of ACTS and can result in specification of verb behavior; modifiers of nominal are PA's (Picture Aiders) and are attributes of objects and may be used for variation. PPs are physical objects. Some special cases of PPs are natural forces and three postulated divisions of human memory: the conceptual processor (where conscious thought take place), the Intermediate memory, and the Long Term memory.

The above refer to conceptual properties and not their sentential ones; for it is possible to have a sentence without a verb, or subject, or have an adjective without a noun--this is not possible at the conceptual level. Since the primitives are not words, they reflect a level of thought underlying language rather than the language. Consequently, the representation is language independent. Each of the categories can be tied together by relations of instrumentality or causation, among others. Thus, they can relate(depend) in specified ways to each other. A dependency structure between two items indicates the dependent predicts the existence of the governing item and not vice-versa. Our intuitive feeling is that these structures could be modeled in a multi-dimensional space which reinforces the development of the model here.

A concept-like structure for representation of objects and "object world" has been implemented by Winston [WI79]. In the model, the computer analyzes a sample and produces a description in terms of objects, their properties and relations linking them. Samples that are not examples (near misses)--those that differ in some striking manner, provide the most valuable information. Concepts are linked by A-KIND-OF (AKO) relationships, with INSTANCE being the inverse, see Figure 4-5. The computer then compares the description of the sample against that of the concept as known so far. This comparison would then produce a list of differences, which are then ranked. Identification of an unknown with a particular concept fails when MUST-NOT relations are present; or when MUST- type are absent.

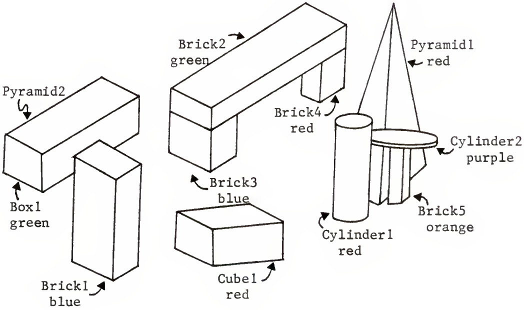


Figure 4-5a. Examples of objects in the blocks world.

```
                     |
       |             | Table
       | Structures--|
       |             | Arch
       |             |
       |
       |             |           |
       |             |           | Pyramid1
       |             | Pyramids--| Pyramid2
       |             |           |
       |             |
       |             |           |
       |             | Cylinders-| Cylinder1
       | Block ------|           | Cylinder2
Things-|             |           |                 |
       |             |                             | Brick1
       |             |                    |        | Brick2
       |             |                    | Bricks-| Brick3
       |             |                    |        | Brick4
       |             |                    |        | Brick5
       |             | Rectangular-Blocks-|        |
       |             |                    |
       |             |                    |        |
       |             |                    |  Cubes-| Cube1
       |             |                    |        |
       |
       | Boxes ------| Box1
```

Figure 4-5b. Organization of the blocks world.

Figure 4-5. The blocks world.

Hierarchical organizations of this type are often used as data structures in computer graphics modeling, as is used in GATORGFX. The use of near misses to facilitate the learning process, again lends itself to the concept space model described.

The use of "time frames" by Winograd [WI72] to solve problems of intent and referencing by anaphora is important to the development of of our model and the application of same will be discussed later. The program uses history of past actions and events to solve problems peculiar to his "world."

CHAPTER 5

A MULTI-DIMENSIONAL CONCEPT SPACE

Basis and Dimension

All of the foregoing models described are characterized by one common feature: they are qualitative, symbolic models, visualized as two-dimensional entities. We however feel that by expanding the dimensionality of the models, beyond the two dimensions, a quantitative depth of understanding will be attained.

In a given vector space V over a field K:

(i) if $u, v \varepsilon V$, $V \neq 0$, then V is closed under addition and scalar multiplication i.e.

$$u, v, \varepsilon V \Rightarrow u + v \varepsilon V$$

$$v \varepsilon V \Rightarrow kv \varepsilon V, \cancel{\forall} k \varepsilon K.$$

(ii) A point in the space V may be uniquely represented by an n-tuple position vector x, where

$$\underline{x} = (x_1, x_2, x_3, . . ., x_{n-1}, x_n)$$

x_i are visualized as being the coordinates of the point in V.

(iii) If $\underline{v}_1, \underline{v}_2, . . ., \underline{v}_s$ are m-component vectors, then the set of all linear combinations of these vectors forms a vector space.

(iv) If a vector space consists of the set of all linear combinations of a finite set of vectors $\underline{v}_1, . . ., \underline{v}_s$

then the vectors are said to span(generate) the space.

(v) The set of linearly independent spanning vectors in a space form the basis for that space, and any two bases of the space contains the same number of vectors, hence it is unique.

(vi) The number of vectors in a basis for a vector space is called the dimension of the space. For a finite number of such vectors, we have a finite-dimensional space.

Theorem 5-1. Given the set $\underline{v}_1, \ldots, \underline{v}_q$ linearly independent set of vectors in a finite-dimensional vector space, there exists vectors $\underline{v}_{q+1}, \ldots, \underline{v}_n \ni$: $\underline{v}_1, \ldots, \underline{v}_n$ forms a basis for the space.

Proof: Let $\underline{u}_1, \ldots, \underline{u}_m$ be the set of spanning vectors in the space (e.g. a basis for the space). Then the set

$\underline{v}_1, \ldots, \underline{v}_q, \underline{u}_1, \ldots, \underline{u}_m$ spans the set.

Since the v's are linearly independent(l.i.) of the preceding vectors in the set, we keep it, otherwise, we exclude it. In this manner, we obtain a set of n vectors that span the space and hence form a basis for V .

Thus if we postulate the existence of a multi-dimensional space, we may iteratively obtain a basis for it in the above manner as long as each successive vector added is linearly independent to the existing members.

Concept Space

Multi-dimensional formalism has been used in the modeling of psychological behavior for it has been recognized that emotional states in posed subjects behave in a strongly multi-dimensional manner [FR72]. These dimensions are roughly defined by their most significant 27 bipolar adjective pairs(e.g. amazed-understanding, complicated-simple). Since the derived meanings of each pair are negatives of each other, we could have 54 adjectives if each adjective is allowed to have its own unipolar meaning representation.

From a linguistic point of view, the meanings of a collection of the 27 adjective features describing the emotional contents present in human faces may be represented as linear combinations of the six orthogonal basis vector found [EA74]. Thus the meanings of each of each adjective word are defined as vectors having six degrees of freedom corresponding to the basis dimensions of the derived emotion space.

The perception of smell and taste has also been shown to behave multi-dimensionally [EA74]. Touch and hearing may be argued to represent temperature and pressure information as functions of space and time. The eyes are receptive to an octave of the electromagnetic spectrum and give information about 3-dimensional positions of objects within their field of view. All of these add up to the argument for multi-dimensionality and also the exploration of a possible basis set for visual information.

In the world of computer graphics, our interest lies in the description of objects, their location, and appearance. The use of adjectives as above, as basis set members allows a description in terms of HEIGHT, DEPTH, WIDTH, and SIZE(not l.i.) in the orthogonal Cartesian coordinate directions (x,y,z). Thus we can then allow the meanings of such words as "short," "tall," "long," "large," "deep," "narrow," "tiny," etc., to be defined vectorially. With the addition of rotational and translational vectors in each of the directions, we can specify modifications to the basic shape.

The medium for the display of graphical entities is a cathode ray tube (CRT) with basic colors RED, GREEN, BLUE, and combinations thereof. Thus the appearance of the object is defined with the addition of these dimensions. To infuse some amount of realism to the overall image, we consider some measure of BRIGHTNESS (not l.i.), TEXTURE, and TRANSPARENCY. These add to the dimensionality of our concept-space to provide various amount of visual features attributable to a given real world object.

Objects

The above dimensions are used to modify the basic description of objects. We now consider the modeling of objects. Most of the physical information about a concept being modeled is carried in the SHAPE dimension; and it is the shape of things that are seen that plays an important part in their conceptualization.

Three dimensional objects are modeled by sets of polygon or patches (cubic-, beta- splines) which describe their exterior surface contours to an acceptable degree of precision. The larger the number of polygons in the description, the more accurately the model can reflect realism.

A polygon is defined to be an ordered matrix of the coordinates of its vertices and we use homogeneous[NE79] four-space representation (x,y,z,w), which allows the translational and perspective information to be expressed in a concise manner. Thus the allowable "grain size" of our formalism is the coordinate. Models may therefore be constructed hierarchically through premultiplication of the primitive polygon matrices by transformation matrices as is shown in Figure 5-1, embodying the correct scaling, rotational, and translational information which moves each primitive to the desired position. Consequently, a cube is just a collection of transformed polygonal matrices, and may be modeled using the data structure of Figure 5-2. This may also be used to generate the entire equivalence class of rectangular polyhedra(BOX, BRICKS,etc.) by incorporating additional rotational, translational and scaling information into the transformation matrix T.

$$\begin{vmatrix} X_1 & X_2 & X_3 \ldots X_m \\ Y_1 & Y_2 & Y_3 \ldots Y_m \\ Z_1 & Z_2 & Z_3 \ldots Z_m \\ W_1 & W_2 & W_3 \ldots W_m \end{vmatrix} \begin{vmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ T_{41} & T_{42} & T_{43} & T_{44} \end{vmatrix} \begin{vmatrix} X_1 & X_2 & X_3 \ldots X_m \\ Y_1 & Y_2 & Y_3 \ldots Y_m \\ Z_1 & Z_2 & Z_3 \ldots Z_m \\ 1 & 1 & 1 \ldots 1 \end{vmatrix}$$

Figure 5-1. Polynomial transformation matrix.

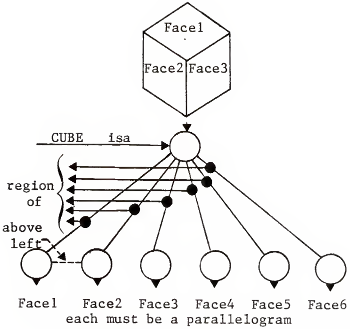


Figure 5-2. Definition of a cube.

This model may be upgraded to represent other features by addition of dimensionality to the point representation used. An example is that of color coordinates. These color coordinates are also dimensional primitives so that the color of a polygon is specified entirely by the transform matrix used. We may use this expansion procedure to incorporate other features, however, in the interest of storage, it is not necessary to store each transform explicitly, since it may be generated when necessary directly from the values of the features which it controls. This is more attuned to the manner in which objects are described verbally and forms the basis for a more formal definition of the concept space.

An OBJECT is defined to be a collection of points in a multi-dimensional feature space in which these points are connected into a hierarchical shape structure. The shape hierarchy for an object such as an arch of Figure 4-5 is shown in Figure 5-3. In such a shape hierarchy, each node would contain a partitioned multispace vector, and only the lowest node would have shape components which were point matrices. At each higher level node, the SHAPE components would contain the complete partitioned vector for each sibling node.

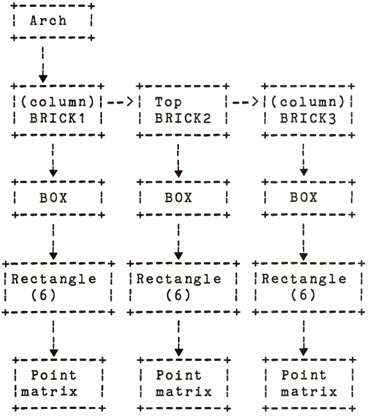


Figure 5-3. Space definition of Arch of Figure 5-5.

An object is therefore defined as an ordered n-tuple, with a set of measures in the various dimensions:

```
OBJECT = [ (SIZE,HEIGHT,WIDTH,DEPTH,ROTVEC(3),LOCVEC(3)),
           COLOR,TRANSPARENCY,TEXTURE,BRIGHTNESS,.......]
```

The first partition in the space, consisting of the first 10 dimensions above, are the shape-affecting dimensions and affect only the physical appearance of the structural relationships implied by the SHAPE hierarchy into which they are embedded. The resulting matrix is diagonal as a result of the linear independence of the concept space dimensions chosen, and allows considerable reduction in computational complexity, since we may now perform four lower order multiplications.

$$\left[\begin{array}{c|c|c|c} A & & & \\ \hline & B & & \\ \hline & & C & \\ \hline & & & D \end{array}\right]$$

A = Shape
B = Brightness
C = Transparency
D = Texture

Figure 5-4. A generalized transformation matrix.

We now define each SHAPE node as being a point in the concept space, and the SHAPE hierarchy as a structure of these points which allows complex OBJECTS to be conceptualized. In the hierarchy of Figure 5-3, each node is associated with an n-space position and feature vectors as described above.

We postulate that the human conceptual model of the world consists of a large number of OBJECTs connected in some kind of relational structures, in a higher order relational space.

The Verbal Transformation

In order to demonstrate the usefulness of the model as presented thus far, we will now explore the transformation from verbal surface structure into the model and its transformation into the visual structure. We will see language in a context as a multi-dimensional phenomenon which probably evolved out of the need of man to communicate information about his environment, which he perceived in a multi-dimensional manner, and used words and group of words as tokens of meaning, since we do not possess any channels capable of directly transmitting some concepts such as smell, tastes, and images. In the need to convert this multi-dimensional information into a form for which we have a channel the language evolved as such a channel. Thus our proposition of Chapter 2 that the producer conceptualizes a multi-dimensional concept to be transmitted, performs a verbal transformation on the concept to reduce it to a sequence of tokens to be transmitted over a low-dimensioned voice channel.

From our definition of OBJECTS to be hierarchical structures of points in a concept space, we note that the displacement of any points in the shape hierarchy will have an effect on the object, and cause a corresponding change in the totality of its conceptualization. The idea of a "short" OBJECT is manifested in its location along the "height" axis of the space. Thus to effect changes in the OBJECTs we can use MODIFIERS to move about the definition points of objects in the space simply by adding displacement vectors of the correct series of modifiers to the object's position.

In English sentences, each noun phrase is visualized as defining an OBJECT(s) with the verbs and prepositions relating or directing their appropriate modifications--a global sequence of operations to perform on the sub-OBJECTS perhaps; and adverbs and adjectives within the noun phrase are seen as modifiers which move the concept space definition of the noun OBJECTS, to create new instances of the OBJECTS. Again, by addition of the displacement vectors of all the modifiers in the noun phrase to the position vectors of the noun.

We need to take cognizance of the fact that certain nouns may not allow some types of modification, however, for there is no such thing as a "tall circle," except wherein we may want to represent an ellipse from a circle with a different major and minor axes. Thus we see each noun as a disjunction of permissible regions in the concept space, each with its nominal vector and SHAPE hierarchy. This process may be guided by the intersection of the noun's feature restrictions with the modifier's displacement vector.

The modifiers may be allowed to have multiple displacement vectors associated with them and the relative magnitudes, where same dimensionality is expressed, need only be consistent in the degree of expression of that dimensionality. Thus any scaling induced by a choice of absolute magnitude will be cancelled by the inverse scaling associated with the inverse verbal transformation. Therefore, the magnitudes of such words as "large" and "huge" would be in ascending order relative to the amount of SIZE associated with each word. The process of parsing a sentence then becomes that of choosing the appropriate meaning vector to apply. The absolute magnitude is unimportant.

The Visual Transformation

The transformation from visual concept model into a picture requires a minimal reduction in dimensionality because of the high bandwidth associated with a picture. One such reduction would be to neglect all but the spatial information along with normalizing the (x,y,z,w) coordinates associated with each polygon in the shape hierarchy. This, however, would not be satisfactory since accurate visual communication will need to entail a hidden surface removal [NE79] process to remove unnecessary tokens and the incorporation of COLOR, BRIGHTNESS, TEXTURE (function of x,y), and TRANSPARENCY peculiar to the concept being displayed. These are sufficient to cue the recognition of the higher dimensional visual concept's although this is not necessary for the exact concept model to have been transmitted. It is only necessary that a concept e.g. BRICK--Figure 5-5--has been conceptualized in the mind of the viewer.

```
   +++++++++++++++++++.
  +++++++++++++++++++..
 +++++++++++++++++++...
+++++++++++++++++++....
!!!!!!!!!!!!!!!!!!!....
!!!!!!!!!!!!!!!!!!!...
!!!!!!!!!!!!!!!!!!!..
!!!!!!!!!!!!!!!!!!!.
```

Figure 5-5. A brick with texture.

In fact, it is our "world knowledge" which associates this figure with a brick, since the figure could be representing an infinite number of real objects. Thus the inverse transformation to a multi-dimensional concept model is complicated by the lack of complete structural information in the tokens transmitted. We suspect that the use of "world knowledge" by humans in the process is tantamount to having a number of OBJECTS conceptualized and a mechanism for comparing the visual projections of the structures with the images seen by the eyes. The area of Scene Analysis addresses this problem and the process then becomes that of moving the OBJECTS in the concept space until the image produced under visual transformation agrees with that seen by the eyes. As a result of the finite number of dimensions in the concept space, adjustments of the transformed image could be carried out one dimension at a time.

Conceptual Structure

The application of this model could be used in the extension of Shank's conceptual structures as detailed in the last chapter. Our model provides a medium for the effective utilization in a quantitative representation of the qualitative information represented in Shank's model. The almost one-to-one correspondence between PP's and PA's of the model and the OBJECT's and MODIFIERs should allow this direct combination. Since all the semantic knowledge about each OBJECT is contained in our concept-space, the PP's and PA's may be merged into that of our OBJECT. The PP's and ACTs are inherently governing categories; and the ACTs primarily involve time-dependent changes in the relationships between various PPs and would thus correspond to time variation of structures of OBJECTs in our concept-space. This could be accommodated in our model by a sequence of relationships to represent time continuum in much the same manner as the individual frames used in animation.

The exact nature of conceptual structures viewed as time varying structures of OBJECTs in our space, nevertheless, requires further study and the decomposition of the verbs which defines the structures into primitive ACTs may yet form a basis for structures in a higher order relational space.

CHAPTER 6

IMPLEMENTATION

We have presented a multi-dimensional concept space in a verbal manner and we shall now discuss a computer implementation of it. As was mentioned earlier, the use of a computer language to communicate a model will not suffer from ambiguity inherent in natural language descriptions. The deterministic nature of computer languages provides a framework in which to verify the completeness and accuracy of a given model. Researchers in AI have long realized the inadequacy of this information transmission medium to accurately communicate complex theoretical models and consequently have adopted a more computational orientation.

In order to substantiate our concept space formalism, a computer program, GATORGFX, was written to implement the basic mechanisms outlined. This program was written in SNOBOL and runs on an IBM 4341 mainframe. The principal mechanism in the understanding process is the parser whose responsibility is to perform inverse verbal transformation from surface structure input into the particular conceptual formalism used--e.g. deep, deep case, annotated surface, conceptual, concept space structures.

Some existing parsers generally view words in the surface string as operands to be manipulated as data. This mode of relationship requires, a priori, that a great deal of language control information be built into the parser; and indeed, such pertinent information is usually stored in a lexicon of individual vocabulary items. This information is usually in terms of binary symbols representing the various set membership criteria which is then used as control by the parser. This schema, may make the design rather complex and the parser inflexible, as further extensions to its understanding abilities may require involved drastic revisions to its mechanics of operation with the attendant growth in size and comprehension.

It may be questionable if indeed the parsing mechanism used by humans is as complicated--since children learn languages without much effort and some adults are proficient in several languages. If such were the case, then our ability to acquire languages may be more difficult.

A more efficient mechanism would be one that is simple, and uses the words in the surface structure as operators to direct the parsing operation. Such mechanism would free the parser from language-dependent control information and obtain its control from the definitions of the words as stored in the lexicon. One avenue of approach to incorporate this type of control would be by addition of structure to the lexicon.

The ability of our program to understand English language imperative and interrogative sentences is illustrated below.

The Parser

We have accommodated this type of control in the data base by storing words and their meanings in a list-structured data base, the Structured Lexicon. These include complete items (like idioms) that are single units at one level of analysis while being made up of several units at another. Thus the associative data base is not limited to single words. These multiple word concepts, however, may be stored as a single concept and assigned a syntactic class based on its total function in a discourse. A heuristic applied to the associative search process attempts to perform a FIRST(x) and FOLLOW(x) operations by finding the longest substring beginning with the first word in the remaining input sequence which has a desired feature. This is done by examining multiple-word links of the lexicon in the data base entries, for the first word in the substring. As an example, the entry for the word "to" contains a multiple-word link (@4PREP). This is an indication to the search mechanism that a four-word expression with a FIRST(word) "to" exists within the data base and has the feature {prep}. Thus in the processing of a sequence ". . .TO THE RIGHT OF A RED BOX" for an instance of a sequence with the feature {prep}, it will first perform an associative search on the four-word sequence "TO THE RIGHT OF." The instance of this subsequence is defined to have a distinct meaning, and hence this entire subsequence is consumed from the input and its meaning from its definition as a four-word group is used as the value in the node which instigated the search. This multi-word associative ability models human language use and processing as we mentioned earlier. From a pragmatic point of view, it is much easier to parse at a idiomatic level than at the single-word level since the associated grammar is much simplified. A chart of the parsing algorithm is given in Figure 6-1 and Figure 6-2 shows sample structured lexicon for a variety of simple interrogative and imperative sentences.

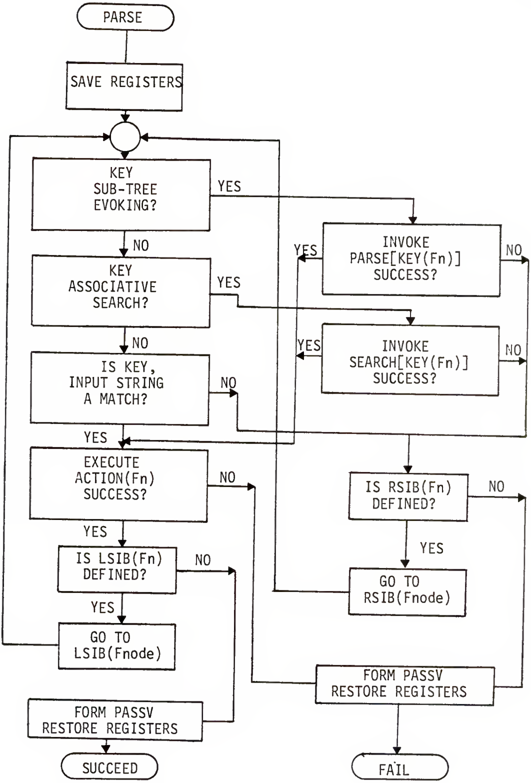


Figure 6-1. Parsing algorithm.

```
    S                                        TO

    |                                         |
+-------+   +-------+   +-------+         +-------+
|  SI   |-->|  SQ   |-->|  SD   |         | {to}  |
+-------+   +-------+   +-------+         +-------+
    |           |           |                 |
+-------+   +-------+   +-------+         +-------+   +--------+
|   1   |   |   2   |   |   3   |         |  IT   |-->|(x,y,z) |
+-------+   +-------+   +-------+         +-------+   +--------+
                                              |           |
                                          +-------+   +-------+
                                          |   4   |   |   5   |
                                          +-------+   +-------+


    NP                                       AT

    |                                         |
+-------+   +-------+                     +-------+
| {npr} |-->| {det} |                     | {at}  |
+-------+   +-------+                     +-------+
    |           |                             |
+-------+   +-------+                     +-------+   +--------+
|  11   |   |  MOD  |                     |  IT   |-->|(x,y,z) |
+-------+   +-------+                     +-------+   +--------+
                |                             |           |
            +-------+                     +-------+   +-------+
            |{noun} |                     |   8   |   |   9   |
            +-------+                     +-------+   +-------+
                |
            +-------+
            |   7   |
            +-------+
```

Figure 6-2a. Stuctured Lexicon

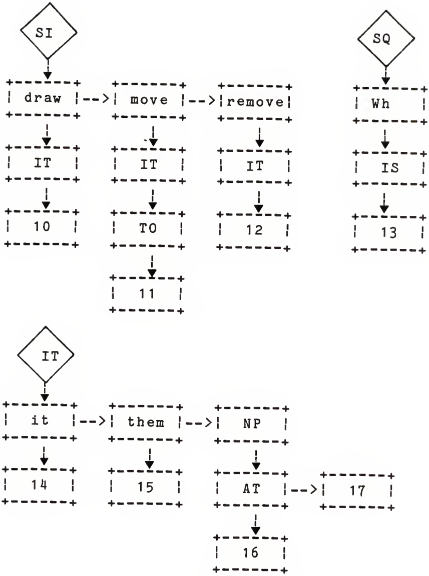


Figure 6-2b. Structured lexicon

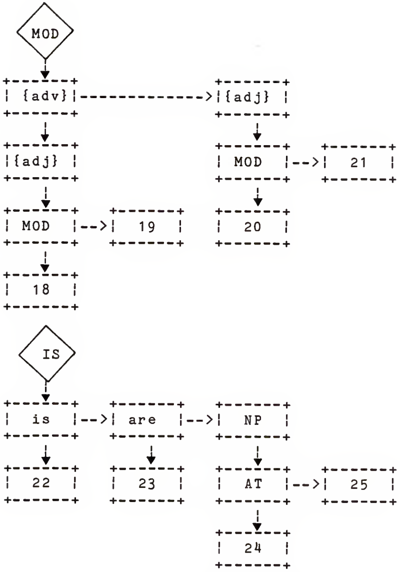


Figure 6-2c. Structured lexicon

Figure 6-2. Structured lexicon.

The structured lexicon differs from the standard in that at any given point in the parsing, only a subset of the words in it are accessible. The parser is an ATN with simple programmer defined functions representing each level of the network of Chapter 4. The main responsibility of the parser is to provide a guide to the syntactic understanding of an input sentence in a top-down fashion, as in most modern language recognizers, by predicting the syntactic ordering of the sentence.

Information is stored in the structured lexicon(SL) in an ad hoc aggregation of list structures defined by Feature Nodes(FNodes), each containing a KEY, VALUE, and an ACTION. Parsing an input sentence involves entering the sentence list structure S and attempting to find a path to a terminal node such that the KEYs of the intermediate FNodes (in certain instances) define the sequence. When a FNode is reached whose KEY can be satisfied by the next input element(s) of the string, the(se) word(s) are removed and the subroutine calls indicated by the node's ACTION are evoked using the node's VALUE as arguments. If the ACTION routine is successful (PASS), parsing continues at the leftmost sibling node in the current structure, otherwise, parsing fails with an appropriate error indicated.

There are two additional KEYs provided to allow the SL possess a greater generative power than that of finite state machine implied by the use of pattern matching. The first is a recursive type which allows the state of the parsing to be suspended and saved on a stack while processing resumes at another FNode of the SL. When the parser reaches a terminal node of the SL, the current state of the parse is reformatted into a concept-space PASS vector (PASSV), and the last parsing context is restored from the stack. This recursive ability increases the generative power to the context-free level, and furthermore, the context-sensitive nature of the list structure makes the SL a context-sensitive formalism. Parsing is successful when the terminal node in the TOP level list structure is reached and the input sequence reduced to the null string.

The second KEY is introduced for computational efficiency. This associative KEY causes the word in the input string to be examined for a particular feature as contained in the data base. It is thus unnecessary for the parser to perform a linear search of all the adjectives, for instance, at every point in the parsing when the sublist ADJECTIVE may qualify a FNode KEY. Rather, the associative data base may be queried in an attempt to determine if the next word in the remaining string has this feature {adj}. When it does, the VALUE returned by the associative search is then used as an argument to the ACTION directed by the FNode evoking the search.

These KEYs are indicated in Figure 6-2 where the elements in the rhombus are the list-head nodes for the various tree-connected FNodes. The words in capital letters indicate recursive parsing at the appropriate list-head nodes, those in lower case indicate pattern matching and the ones in curly braces are used in the associative search for "feature." The VALUEs and ACTION of the FNodes are not shown. While this representation may not be as concise as the ATN due to the redundancy introduced by its tree structure, it is nonetheless any less powerful in its description. The formalism is prudent, and exhibits the positive attributes of structured programming.

Representation of Meaning

The values stored explicitly within the FNodes and implicitly within the associative data base are used to represent portion of the word meaning which is representable as vectors in the concept space. The remainder of the meaning attributable to a word or class of words is represented in terms of the ACTIONs contained in the various FNodes. Also, word meanings are defined by their effect upon the conceptualization of the sentence in which they are resident. [SH73] in a classification of word meaning has defined the following classes of meaning (i) words which perform an action, (ii) words which modify an action, (iii) words which produce a mental picture, (iv) words which modify a mental picture. Words in types (i) and (ii) incorporate their meanings in the ACTIONs within the SL while those in (iii) and (iv) contain most of their meanings in their VALUES. This is not meant to be a hard and fast dichotomy, since part of the meaning of every word is its effect on the language-independent algorithm.

We have used a set of registers(26)--Table 6-1--in the algorithm, in which concept space meanings are processed, and the register context is automatically saved on and restored from the stack which is used for recursive control. The context is saved in a sparse vector format for efficiency and upon popping, a PASSV, which contains the terminal state of the lower level structure, is available to the calling FNode. The PASSV is formatted as

PASSV = [R*V/]+ where

R = register name (value of which may not be null)
* = an operation to be performed (addition(+), multiplication(.), exponentiation(..), replacement(<-))
V = value associated with the vector coordinate

Table 6-1. Register dimension for concept space.

REGISTER	DIMENSION	REGISTER	DIMENSION
A	DEFINITENESS	K	GREENESS
B	NUMBER	L	BLUENESS
C	WIDTH	M	INTENSITY
D	HEIGHT	N	ACTION
E	DEPTH	P	X-ROTATION
F	SIZE	Q	Y-ROTATION
G	SHAPE	R	Z-ROTATION
H	TRANSPARENCY	X	X-COORD
I	TEXTURE	Y	Y_COORD
J	REDNESS	Z	Z-COORD

The concept space value of the words are identical to the format of the PASSV, thus not only is the value of each vector coordinate specified, but also the effect which it must also have on the register. Meaning vectors are assimilated into the register context by a DECODE function which performs anomaly checking by using special markers which may be placed in the registers to assure subject-verb agreement, number and feature agreement etc. Execution of the DECODE function is one of the ACTIONs which may be invoked by a FNode and its success results in the success of the sub-tree under evaluation. The interaction of a given vector component with its associated register is facilitated by a merger table of Table 6-2.

Table 6-2. Merger table.

		MEANING VECTOR(M)			
		null	+	-	other
REGISTER (R)	null	R <- M	R <- M	R <- M	R <- M
	+	R <- R	R <- R	FAIL	R <- M
	-	R <- R	FAIL	R <- R	FAIL
	other	R <- R	R <- R	FAIL	R <- R*M

Decoding

When a concept with a desired part-of-speech has been parsed, a relationship vector (RELVEC) is returned which must then be decoded onto a set of local and global registers under control of the parser. Upon popping to a calling routine in the parser, the resultant state of the local registers is reformatted as a higher level concept which can then be decoded as a single concept by the calling routine.

When particular attributes e.g. size, degree of color, are not specified, the concept OBJECT implies a default value for the corresponding registers. For example, the concept "very tall" is converted into a numerical value twice the height of the OBJECT. This function "very tall," is interpreted as a set of operators on the height and hence is relative only to the height dimension of an object. Tallness thus, is relative to an implied norm value. Modifying words such as these are retained as relational operators on the HEIGHT dimension of the image in the Short Term Memory.

The importance of this being that on concepts such as this, the system is provided with a numerical value for height that can then be mathematically compared with other height concepts. Thus if "enormous" has been defined to be larger than "large" conceptually, an enormous object could be mathematically inferred to be large also.

Execution

Merely representing an object is not sufficient for verbal communication with the graphics system. The system has to understand the intentions inherent in the conversation with regard to the OBJECT specified. Verbs and prepositions have been classified as RELATIONal words which are manipulators of objects. The conceptual actions implied by these words are expressed by a set of executable programs which use as input images produced by the input sequence and the memory of the images being displayed. These routines control and induce decode, create, seek, compare, change, and remove functions for the objects. Each Relation places in register N a set of program names which are executed on the completion of parsing at the level on which they were parsed.

Memory

The memory of our system consists of both long-term (LTM) (in the form of a dictionary), and a short-term (STM) memory in the form of a description of the objects being displayed. Both types of memory are called into use in the understanding of verbal communication with the graphics system. LTM is responsible for storing the "truths" about the concepts as applied to the graphics system. It describes the scope and limitations of a concept in general and indicates its relationship to other concepts in the form of a multi-dimensional vector. The STM consists of the list of objects currently being displayed by the system. These are resultant OBJECT concepts generated by the manipulation and merging of the multi-dimensional vectors of the concepts expressed in commands and communication to the system, and are stored in the same notation as the RELVEC entry of a dictionary. The information found in an entry of this memory, therefore, retains only the effects of the modifying concepts and relations concepts of the image.

Interactive Dictionary

A new multiple or single word OBJECT or MODIFIER concept can be defined on-line by expressing the definition of the concept in terms of previously defined concepts. The Dictionary Entry Routine (DER) passes the definition of the new concept to the parser which in turn calls DECODE and possibly EXECUTE in order to resolve and interpret the higher level concept being defined. Anomalous and ungrammatical phrases are recognized and are rejected. Once the higher level is determined, DER sets the appropriate linkages in the dictionary for the multiple word concept and assigns the correct attributes to the new entry.

An Example

Operation of the parser is dependent on the use of the multi-dimensional representation of meaning presented. Parsing a sentence is accomplished by a call to the function PARSE(S,STRING), where S is the pointer to the sentence tree, and STRING is the input string. Let us consider the sentence:

DRAW A VERY TALL RED BRICK

The operation of the parser is initiated by a call to parse at the tree S. The first FNode has the KEY=SI and this results in a recursive call to the corresponding tree. The entire meaning of the input "draw" is contained in the ACTIONS of the nodes which succeed its node in the SI structure. The only function of this word is to select the correct branch of the SI tree to process. Parsing at this level consumes the word DRAW and evokes a recursive call to the IT tree, which evokes another call to the NP tree since the KEYs "it" and "them" fail in the FOLLOW(x) match to the remaining string.

In the NP network, the feature {npr} fails but the feature {det} succeeds. The determiner "a" is parsed and its relationship vector would cause the register A (definiteness register) to be set to indefinite (-), and a number agreement test to be performed on the register B which subsequently is set to indicate one object. The meaning of vector stored for "a" in the associative data base is

a = @1D A<- -/B<- 1/$

This indicates a single word group with feature {det}, and concept space meaning indefinite singular. The meaning vector is used as the argument to the decode function--the specified action for the node. Decoding of the relationship vector onto NP registers yield:

A = '-'; B = '-'

Parsing then continues with a call to the MOD subnetwork. This sets all the MOD registers to null. In this subnetwork, the string "VERY" is noted to be an {adv} and the ACTION at the node is to save its meaning vector in a temporary storage location. The next string is consumed and the ACTION associated with the "TALL" {adj} specifies that its meaning be decoded onto the MOD registers which in this case results in

D = 1

At this point another operation specified by the ACTION causes the saved vector corresponding to {adv} to be DECODED. A multiplication factor is specified and its entry yields

D = 2

Reformatting the MOD registers results in

MOD = CJ/DJ/EJEG2#EF2#/FJ/MJ/

Another recursive call to the subnetwork MOD causes the removal of the string "RED" and

J = 1

A subsequent call to the MOD network fails since the next input string does not have the appropriate feature. A pop to the calling routine then ensues after the reformatting of the J register into the PASS vector

PASSV = J + 1/

and a decoding of the relationship vector onto the MOD registers. Thus

D = 2; J = 1

This successful return from the subnetwork leaves a composite pass vector

PASSV = D + 2/J + 1

The MOD node of the NP network succeeds and the PASSV is decoded onto its registers:

A=-; B=-; D=2; J=1

Next the string "BRICK" is parsed and the meaning vector decoded onto the NP registers:

A=-; B=-; D=2; G=$BRICK

It is important to note that had we indicated

*DRAW A VERY TALL BRICKS

The DECODE function would have failed since "BRICKS," has a plural feature (+); but the (+) and (-) flags cannot be combined as indicated in the table. The successful traversal of the NP network causes a successful termination with the pass vector returned

PASSV = A+-/B+-/D+2/G<- $BRICK/J+1/

Parsing then resumes at the IT network and the ACTION induced has the PASSV decoded onto the IT registers.

In a similar vein to the above, the sentence "DRAW A VERY TALL CIRCLE" would fail since the noun circle even though has a number agreement with the {det}, contains "negative" flags in the height, width, and depth dimensions.

Time frame

The ACTION taken at the IT node is to create a program which, when executed, will cause an update of the current time frame to include the addition of new OBJECTS, which may have been created. A similar program will allow the current time to be restored to its previous state when executed. The program is also executed by the SI node and becomes a permanent fixture in the data base.

Thus the program maintains a series of programs to move the current time frame forward in time, and another set to move it back in time. This capability allows the correction of incorrect program responses and permits animation sequences to be generated. Presently, there is no facility for the insertion of additional--in between--time frames to those already in place, although the extension is straightforward.

We shall now complete the illustration given above by including this use of time frame. Suppose the sentence had been "DRAW A VERY TALL RED BRICK AT THE GREEN CIRCLE." Transfer of parsing to the AT node consumes the string "AT" and eventual recursive call to the NP network, which parses "the green circle" and returns a vector which conceptualizes the string. The action of the NP node of the AT network yields

PASSV = A++/B+-/G<- $CIRCLE/L+1/

to be decoded and the OBJECTS existing in the current time frame is searched for an instance and location of "THE GREEN CIRCLE." To achieve this, the registers pertaining to the AT tree, having no null dimensions, are compared with each OBJECT in the current time frame, and a list is compiled for all the OBJECTs with "shape" CIRCLE and "greenness" greater than or equal to unity. This of course does not preclude circles with values two or greater if they existed.

After a compilation of this list, a check is made for "number" and "definiteness" agreement with the values in the AT registers. If a definite (+) is found, then the number of OBJECTs found is required to agree exactly with the contents of the B register; otherwise, a (-) flag will confirm agreement provided the number of objects located is non-zero. A green circle found located at say (1,2,3) in the concept space of the current time frame would then yield a vector

```
PASSV = X+1/Y+2/Z+3
```

and the ACTION of the node AT of the IT tree will cause the vector to be decoded onto its registers and thus successfully terminate, returning a vector to the IT node of the SI to represent a concept space conceptualization of

```
DRAW A VERY TALL RED BRICK AT (1,2,3)
PASSV = A+-/B+-/D+2/G<- $BRICK/J+1/X+1/Y+2/Z+3/
```

The above illustrates an aspect of the mechanism employed and the topology of adopted for the SL. If a network topology had been used, instead of this hierarchical topology, the kind of selection process provided to the appropriate branch of the SI tree might not have been possible.

Graphics Display System

While one of the functions of the parser is to use the natural language input commands of a user to maintain a current time frame containing the conceptualizations of objects which have been created, the function of the display subsystem is to use the contents of such time frame to produce a color television pictures of objects described therein. This subsystem is not quite operational at this point, but the capabilities of the program have been tested and validated on an IBM 4341 mainframe. We hope, with the imminent commissioning of the new CAD/CAM facility of the College of Engineering, to transport this program to this IBM mainframe.

When it is all completed, we hope to install a set of function switches as analog input dials for interactive specification of the VALUES (COORDLOC(3), COORDROT(x,y), SIZE) of the dimension and COLOR dimension in the limited concept space.

CHAPTER 7

FUTURE CONSIDERATIONS

Before the invention of the digital computer, language was seen as the exclusive province of humans. With the advent of the computer, new possibilities appeared. It was recognized that in addition to their obvious applications in the scientific disciplines and bookkeeping, computers could work with language.

Question Answering

Most natural language understanding system demonstrate their capabilities by answering questions about facts in their database. Systems can be characterized by the nature of their data bases as being text based or encoded systems. In question answering, the system attempts to generate answers to specific questions based on stored information. In these systems, declarative sentences are often input first, and a data base of facts constructed using the information contained in the sentences. Question answering has been the central project in the development of computational model of language.

In GATORGFX, the ability to answer questions is somewhat limited. The system differs from most natural language systems in that its degree of competence is demonstrated visually rather than verbally. Thus this ability is not germane to effective communication with the system. The inverse transformation needed to convert to surface format from a conceptual model in which information is stored, is complicated by the fact that all surface string words are removed in the encoding process and thus cannot fully guide the decoding process.

Questions may be divided roughly into two categories depending on the degree of response desired. There are those to which a simple Yes/No response is all that is required. The response to "IS THE BIG BLUE BRICK AT (X,Y,Z) LARGER THAN THE HUGE RED BOX?" is a simple Yes/No.

As we mentioned in the earlier chapters, the existence of distinct syntactic structures for interrogative sentences allow their easy discrimination from those such as declaratives and imperatives. These are typified by the letters "wh-." GATORGFX is capable of handling some of these wh-question types (What, Where, How) using the SQ subnetwork, along with the SI tree.

A sentence such as "CAN YOU DRAW A TALL RED CIRCLE?" requires the program to evaluate its own processing abilities and may be answered without storing the information explicitly. In this case the SQ tree parses "can you" and the SI tree can then be invoked. Of course, in this example the response would be "no" since the NP network would find tall and circle incompatible. A successful evaluation should prompt the "yes" response.

WHY DID YOU DRAW THAT? This requires an examination of the motivation behind each action which the program undertook. Since the only motivation is normative, the response is "...BECAUSE YOU TOLD ME TO."

WHAT IS ABOVE THE GREEN BRICK? Here, the response is generated from the list of the OBJECTs in the current time frame which are located above the "green brick" if one exists. This involves the enumerating of the concept space coordinates of all OBJECTs. A better scheme toward an understandable response may be to implement an inverse verbal transformation. All word meaning vectors could be linked in a sorted list by their dimensionality. Thus faced with an OBJECT which contained size, blueness, redness, $Brick dimensionality, we may determine by examination of the list that the words "fat purple brick," when parsed would result in a similar conceptualization.

Declaratives

A "good" natural language system should be able to use the information in declarative sentences to expand its descriptive capabilities or allow more subtlety in expression. Declaratives provide a means for increasing the scope in which a program is capable of interacting. Thus, we need to define a method for generalizing our concept models. There is a definition for a concept such as a "brick" but it would be splendid to generalize its definition by presentation of other examples and non examples much like in Winston's [WI75,79] methodology. Since our concepts are rich in dimensionality, generalizations may be made modeling human concept formalism. Our program accepts limited subset of declarative construction which may be used on-line to define new words to the system. These new word meanings are formalized within the existing dimensionality of the system. This affects the subtlety of expression to which the system is capable of responding. Ability to cause addition of other dimension to the space, should extend the descriptive power of the system.

A declarative like "Nell drew a tall box" would require the conceptualization of Nell(an object) to that of "drawing" an object. As a crude approximation, we may maintain a separate "time quantum" for every active object in the system's memory. This should allow for modeling of conceptualizations involving time using Shank's type Conceptual Structure; with verb meanings modeled as the OBJECTs are presently, using the dimensionality of Case Structures(Tense, Mood, etc.), and the SHAPE being the ACTs.

Non-Visual Dimensions

One easy extension may be the addition of non-visual dimension to the concept space e.g. taste(sweet,salty); smell(burned, flowery). Thus, it would be possible to conceptualize "A HERO SANDWICH." We would, however, be faced with an avenue to verify the accuracy. Thus, we can do no more than rely on verbal comparisons between OBJECT's flavors and on the structural consistency imposed by the visual portions of the concept space model.

All of these new dimensions to the concept space can be created on-line at execution time by additions of new registers in the parser. The language allows the creation of new variables at execution time and this could be accomplished by addition of new words expressing the added dimensionality to the data base.

Speech Recognition

Efforts in the area of speech processing have had considerable success using predictive parsing techniques of ATN. Miller[MI74] uses an ATN grammar in a configuration such that parsing may commence in the middle of a string, "island of reliability" and progresses both leftward and rightward. Simultaneous processing of several such "islands" may be allowed as a threshold is lowered until a complete parsing results. This requires that subnetworks in the grammar contain pointers to the contexts which may evoke them.

The application of the SL to speech understanding may result in eliminating the associative KEY type. Since this was introduced to enhance searching for strings exhibiting a common syntactic features, by reducing the pattern matching to a phoneme level, should reduce the number of searches. It is thus not necessary to perform an associative search for "noun" rather, parsing would be evoked for the NOUN subtree in which all the nouns have been stored by phonemes. The meaning attached to each word may be stored in the terminal node, and parsing may proceed in a "first best" manner based on some probability distribution of the phoneme derived from the phoneme recognizer.

Computer Aided Instruction

Computers are being used for education in various ways, often involving some kind of question-answer interaction. For some domains, the interaction may be in a coded symbolic form. For most areas, it is natural for questions, answers, and discussions to be in natural language. Most existing systems rely on "canned" language in the form of questions, answers, and explanations that are pre-planned. A more effective teaching program could be built if the computer could deal with both the content of pre-stored material and the user's queries and responses. Systems that integrate natural language with other uses--graphics, simulation--in teaching, have been built. We think, that with the addition of other capabilities, as above, our system can aid in this type of instruction.

Conclusion

A multi-dimensional concept space has been developed in which conceptualizations of real world objects exists as hierarchical structure of points. The formalism uses the linear independence of features usually used to describe real objects to model the objects in this space. The resulting hierarchical structure was shown to be compatible with current Conceptual Theory in that a transformation is shown whereby PPs do produce pictures. The objects are similar to the PP's and are based on primitive SHAPEs. While the class of objects using the model may be concrete, real physical entities, the conceptualization to some abstract entities has been mentioned.

We have demonstrated that there exist at least a conceptual model which relates visual and verbal concepts within a common structure. This system employs heuristics in concert with "meta-knowledge." The fact that there is a multi-dimensional model reinforces our intuitive feelings about the multi-dimensional nature of the human cognitive process.

The formalism has been implemented as a computer graphics command language, which uses a Structured Lexicon, to form a multi-space conceptualizations from the input sentences. The concept space formalism has been shown to be compatible with accepted linguistic theories and tenets, as is the parsing technique.

CHAPTER 8

RESULTS

The program GATORGFX allows English language specification of objects to be modeled on a color graphics facility of the Computer Graphics Research Laboratory(CGRL) at the University of Florida. In a full fledged operation, the program will accept simple English commands and translate them into the conceptual space model and issue appropriate subroutine calls to the graphics system to display the structures thus constructed.

The following illustrates the level of sentence understanding exhibited by this program. The program should generate three basic commands to the graphics subsystem. They are *LOAD(), *MODIFY(), *DELETE(). All the statements prefixed with an asterisk is transparent the user, so that the only response observed is: I UNDERSTAND. Those prefixed by "#" are sentences which are "ungrammatical" to the system in the course of the dialogue we are conducting with the system. For sentences which fail to be parsed, the response is a # followed by the sentence which could not be analyzed, followed by a message about why the failure occured and then followed by: I DON'T UNDERSTAND. The arguments generated by the program for the commands are numerical values representing the logarithm, or actual value of the feature, whichever is appropriate.

A typical order is

```
*LOAD(object number,shape,log width,log height,
     log depth, log size,log transparency index,
     log texture index,redness,greenness,blueness,
     brightness, x rotation,y rotation,z rotation,
     x location, y location,z location)
```

Currently, GATORGFX recognizes only two types of nouns--BOXES or any structurally similar objects(BRICKS, CUBE etc.) and CIRCLES--however, other nouns may be added easily to the lexicon. With these primitive shapes stored in the graphics subsystem, more complicated structures can be defined in terms of these. Exhaustive error checking is performed to aid in the determination of correct meaning.

A "load" command is generated whenever an imperative sentence is recognized which contains the verb "draw" or its synonym. We will now run through a dialogue with GATORGFX. The machine's response to various commands is given:

(i) DRAW A LARGE RED CIRCLE AT (1,2,3)

is

```
*LOAD(1,CIRCLE,-,-,-,1, ,0,1,0,0, , , , ,1,2,3)
I UNDERSTAND
```

(ii) #MOVE A CIRCLES TO (4,5,6)

```
THE NUMBER AND FEATURE OF "A CIRCLES" IS ANOMALOUS
I DON'T UNDERSTAND
```

Here, the determiner "a" expects the feature of the noun to be singular. Since the number feature of the words do not agree the system rejects the sentence.

(iii) #MOVE 2 CIRCLES TO (4,5,6)

```
THERE IS ONLY ONE FIGURE OF THAT DESCRIPTION
```

```
      I DON'T UNDERSTAND
```

Although, the determiner "2" and the noun "CIRCLES" are in agreement, inspection of the data base to locate the objects fails since only one such object is present [namely a RED CIRCLE at (1,2,3)], and the user is so advised.

```
(iv)  #MOVE THE BLUE CIRCLE TO (4,5,6)
      THERE ARE 0 FIGURES OF THAT DESCRIPTION
      I DON'T UNDERSTAND
```

Here, examination of the data base yields only a red circle and the operator is advised.

```
(v)   MOVE THE LARGE CIRCLE TO (4,5,6)
      *MODIFY(1,CIRCLE,-,-,-,1, ,0,1,0,0, , , , ,4,5,6)
      I UNDERSTAND
```

In this case the large circle is located in the data base. The command involves the modification of the existing object.

```
(vi)  MOVE IT TO (6,7,8)
      *MODIFY(1,CIRCLE,-,-,-,1, ,0,1,0,0, , , , ,6,7,8)
      I UNDERSTAND
```

This is an example of an anaphoric reference. With this feature the system is required to remember which objects were processed or referenced in the prior frame. Since the system keeps a complete time history of events, errors which may be due to interpretation of user's intent is easily corrected.

```
(vii) #MAKE THEM LARGE
      "THEM" IS UNDEFINED
      I DON'T UNDERSTAND
```

This is an incorrect use of the above feature, since no plural objects have been discussed yet, the system rejects the command.

```
(viii) #MAKE IT VERY TALL
       THE HEIGHT FEATURE OF "MAKE IT VERY TALL"
        IS ANOMALOUS
       I DON'T UNDERSTAND
```

In this instance, the referent is to the last circle modified and is understood by the system. However, as we mentioned in Chapter 5, the program allows nouns to reject certain modifiers or adjectives. In this case, the dictionary entry for circle disavows any specification for height, width, or depth(a circle is two dimensional) since a "tall circle" becomes an ellipse. Thus, the "make it tall" is rejected.

```
(ix)  MAKE IT MUCH MORE TRANSPARENT AND ROUGH
      *MODIFY(1,CIRCLE,-,-,-, ,1,1,1,0,0, , , , ,6,7,8)
      I UNDERSTAND
```

As in the above, modifications which are allowed in the noun-modifier restrictions may be compounded. Here, "much" is assumed to modify both "more transparent" and "rough." If this interpretation is not satisfactory the user has the option to cause the system to back up over the "offending" command and replace it with a new one.

```
(x)   NO,
      MAKE IT MUCH MORE TRANSPARENT AND A TEENY BIT ROUGH
```

```
*MODIFY(1,CIRCLE,-,-,-, , ,1,1,0,0, , , , ,6,7,8)
*MODIFY(1,CIRCLE,-,-,-, ,1,1.049999,1,0,0, , ,
                                         , ,6,7,8)

I UNDERSTAND
```

The prefix "no" indicates to the system to step back in time prior to the preceding instruction, before executing the indicated command. We have approached the treatment of this type of ambiguity rather single-mindedly because of the single processor nature of the implementation. Some other AI languages [BO74], employ multiple processing features and this should handle the type of ambiguity above more naturally.

However, the ability of the program to define multiple word idiom: "more transparent," "teeny bit" as single syntactic units is of interest. Features such as these, models the human speaker, in situations where the total sum of meanings of the constituent words of the idioms does not equal the total meaning of the sum of the words.

(xi) COLOR IT A LITTLE BIT REDDER

```
*MODIFY(1,CIRCLE,-,-,-, ,1,1.049999,1.25,0,0, ,
                                     , , ,6,7,8)

I UNDERSTAND
```

Here, again, the system is required to remember objects referenced in prior frame. Thus, the "very transparent and rough" circle made 25% "redder." This also illustrates use of the idiom feature "little bit."

```
(xii) NO,
      COLOR IT PURPLE
      *MODIFY(1,CIRCLE,-,-,-, ,1,1.049999,1,0,0, ,
                                       , , ,6,7,8)
      *MODIFY(1,CIRCLE,-,-,-, ,1,1.049999,1,0,1, ,
                                       , , ,6,7,8)
      I UNDERSTAND
```

Again this illustrates the backup feature. We note, that the "no" restores the "much more transparent" and "teeny bit rough" circle.

```
(xiii) XROTATE IT 10 DEGREES AND YROTATE IT 20
        DEGREES
      *MODIFY(1,CIRCLE,-,-,-, ,1,1.049999,1,0,0, ,
                                     10, , ,6,7,8)
      *MODIFY(1,CIRCLE,-,-,-, ,1,1.049999,1,0,0, ,
                                    10,20, ,6,7,8)
       I UNDERSTAND
```

This specifies the ability of object rotations.

```
(xiv)  DRAW 3 VERY SMOOTH SMALL AND VERY OPAQUE CUBES
      *LOAD(2,CUBE, ,-1, , ,-2,-2, , , , , , , , , , )
      *LOAD(3,CUBE, ,-1, , ,-2,-2, , , , , , , , , , )
      *LOAD(4,CUBE, ,-1, , ,-2,-2, , , , , , , , , , )
      I UNDERSTAND
```

This allows the creation and modification of multiple objects within the same command.

```
(xv)  MOVE 2 OF THEM TO (8,9,10)
```

```
      *MODIFY(2,CUBE, ,-1, , ,-2,-2, , , , , , , ,8,9,10)
      *MODIFY(3,CUBE, ,-1, , ,-2,-2, , , , , , , ,8,9,10)
      I UNDERSTAND
```

Here we illustrate the use of complicated references. Note that the first two were selected.

```
(xvi) MAKE THEM MORE TRANSPARENT THAN THE PURPLE CIRCLE
      AT (6,7,8)
      *MODIFY(2,CUBE, ,-1, , ,1.5,-2, , , , , , , ,8,9,10)
      *MODIFY(3,CUBE, ,-1, , ,1.5,-2, , , , , , , ,8,9,10)
      I UNDERSTAND
```

This illustrates modification relative to other existing objects. We note that "more transparent" yields a factor of 50%

```
(xvii) #MAKE THEM AS ROUGH AS THE BLUE CIRCLE
       THERE ARE 0 FIGURES OF THAT DESCRIPTION
       I DON'T UNDERSTAND
```

Here, no "blue circle" has been defined prior to the current time frame and the system responds accordingly. The system checks to make sure that only sentences which make sense are executed.

```
(xviii) #COLOR 4 OF THEM BLUE GREEN
        THERE AREN'T THAT MANY OF THEM
        I DON'T UNDERSTAND
```

This is similar to the above.

```
(xix) MAKE 1 OF THEM AS ROUGH AS THE PURPLE CIRCLE
      AND MAKE THE REST OF THEM SMOOTH
      *MODIFY(2,CUBE, ,-1, , ,1.5,1.049999, , , , ,
                                       , , ,8,9,10)
      *MODIFY(3,CUBE, ,-1, , ,1.5,-1, , , , , , , ,
                                                 8,9,10)

      I UNDERSTAND
```

Here the system "remembers" that "them" refers to CUBES. The first is made rough, the second smooth. This also illustrates use of compound sentences so a user can make use of the features that we have illustrated.

```
(xx)  REMOVE THE CUBE AT (8,9,10)

      *DELETE(3)

      I UNDERSTAND
```

We illustrate here, a command for the elimination of objects which are no longer of interest to the user. These may be deleted provided they exist; else the user will be notified accordingly. The first of the two cubes at (8,9,10) is removed.

```
(xxi) #REMOVE THE TALL CUBE AT (8,9,10)

      THERE ARE 0 FIGURES OF THAT DESCRIPTION

      I DON'T UNDERSTAND
```

Although there exist a cube at this location, but we did not define it as "tall" in the creation hence the response from the system. (Note: even though a "tall" cube does not make any sense, for the purposes of illustration we are considering a CUBE as a BOX).

```
(xxii) #COLOR A CUBE TALL

       THE HEIGHT FEATURE OF "COLOR A CUBE TALL" DOES

        NOT COMPUTE
```

```
    I DON'T UNDERSTAND
```

Here, extensive checks for semantic agreement of the various portion of an input string rejects a sentence which although is grammatically correct, as anomalous.

```
(xxiii) DRAW A TALL RED CUBE AND A SMALL BRIGHT
          BLUE CIRCLE
          *LOAD(5,CUBE, ,1, , , , ,1,0,0, , , , , , , )
          *LOAD(6,CIRCLE,-,-,-,-1, , ,0,0,1,4, , , , , , )
          I UNDERSTAND
```

In this illustration, the feature values of "small" and "bright" are appropriately reflected.

```
(xxiv)    COLOR A TALL CUBE GREEN AND MOVE IT TO (11,12,13)
          *MODIFY(5,CUBE, ,1, , , , ,0,1,0, , , , , , , )
          *MODIFY(5,CUBE, ,1, , , , ,0,1,0, , , , ,11,12,13)
          I UNDERSTAND
```

The instance of a "tall cube" that already exists is acknowledged, the color is changed and the position modified.

```
(xxv)    DRAW 3 CIRCLES AT (14,15,16) AND COLOR 2 OF THEM RED
         *LOAD(9,CIRCLE,-,-,-, , , , , , , , , , ,14,15,16)
         *LOAD(10,CIRCLE,-,-,-, , , , , , , , , , ,14,15,16)
         *LOAD(11,CIRCLE,-,-,-, , , , , , , , , , ,14,15,16)
         *LOAD(9,CIRCLE,-,-,-, , , , 1,0,0, , , , ,14,15,16)
         *LOAD(10,CIRCLE,-,-,-, , , ,1,0,0, , , , ,14,15,16)
         I UNDERSTAND
```

The first 2 circles are colored red. The following three groups of command illustrate the completeness of the history of a session as maintained by the system. This feature helps the operator to review the development of his image(s) or set up a sequence of frames for animation. A program to generate movies will require interpolation between successive time frames.

```
(xxvi)   GO BACK IN TIME
         *MODIFY(7, , , , , , , , , , , , , , , , , )
         *MODIFY(8, , , , , , , , , , , , , , , , , )
         *DELETE(7)
         *DELETE(8)
         *DELETE(9)
         I UNDERSTAND
(xxvii)  GO BACK IN TIME
         *MODIFY(5,CUBE, ,1, , , , ,1,0,0, , , , , , , )
         *MODIFY(5,CUBE, ,1, , , , ,1,0,0, , , , , , , )
         I UNDERSTAND
(xxviii) GO FORWARD IN TIME
         *MODIFY(5,CUBE, ,1, , , , ,0,1,0, , , , , , , )
         *MODIFY(5,CUBE, ,1, , , , ,0,1,0, , , , ,11,12,13)
         I UNDERSTAND
(xxix)   GO FORWARD IN TIME
         *LOAD(7,CIRCLE,-,-,-, , , , , , , , , , ,14,15,16)
         *LOAD(8,CIRCLE,-,-,-, , , , , , , , , , ,14,15,16)
         *LOAD(9,CIRCLE,-,-,-, , , , , , , , , , ,14,15,16)
         *MODIFY(7,CIRCLE,-,-,-, , , ,1,0,0, , , , ,14,15,16)
         *MODIFY(8,CIRCLE,-,-,-, , , ,1,0,0, , , , ,14,15,16)
         I UNDERSTAND
```

```
(xxx)     #GO FORWARD IN TIME
          ATTEMPT TO GO FORWARD IN TIME FAILED
```

Here an attempt to go into the future is disallowed.

```
(xxxi)    "HUGE" IS VERY LARGE
           I UNDERSTAND
```

This type of communication with the system is an attempt to define new words using combinations of other words since the initial vocabulary defined for the system may be inadequate for some users. The system allows this kind of input defined in terms of words already known by the system. This is facilitated by the multi-dimensional representation of meaning used. It allows word meanings to be "combined" to produce a new meaning representation which has the same vector format as those of the original words. Here, the word "huge" is defined to be equivalent to "very large." This definition creates an initial multi-space meaning for the word "huge" which may later be modified independently by "very" and "large."

```
(xxxii)    "SKY BLUE" IS BLUE AND GREEN
            I UNDERSTAND
```

In a similar manner we have defined "sky blue" as equivalent to equal amount of blue and green.

```
(xxxiii)    DRAW A HUGE SKY BLUE CUBE
            *LOAD(10,CUBE, , , ,2, , , ,1,1,0, , , , , , )
            I UNDERSTAND
```

This demonstrates the ability of the system to use words which were created previously. With this kind of capability it is not difficult to implement a word-modification ability for the system so that "MAKE THE WORD 'SKY BLUE' BLUER" is correctly interpreted.

(xxxiv) GOOD BYE

BYE, SEE YOU LATER

This terminates the session.

GLOSSARY

=>	implies
<=	append to
<-	replace
↑	node from which a recursive call was made
x<-(↑)y	means initialize x to the y of ↑
(x)y	means the y of x
!copy	a copy based on ^

0 = null

# = word boundary

## = sentence boundary

S = sentence

PreS = PreSentence

NP = noun phrase

VP = verb phrase

PP = prepositional phrase

P = preposition

aux = auxiliaries

Adj = adjective

Adv = adverb

mods = modifiers

Imp = imperative

Int = interrogative

Decl = declarative

Inf = infinitive

Poss = possessive

Subj = subjective

Obj = objective

Pro = pronoun

Trans = transitivity

Det = determiner

Pass = passive

Q = question

Det = determiner

M = modal

Past-P = Past-Participle

Pres = Present

Trans = Transitive

Sing = Singular

Plu = Plural

q-e = question element

d-o = direct object

i-o = indirect object

m-v = main verb

```
A
/\ = abbreviation for a string which
--   is dominated by a constituent, A,
     where the detailed constituent
     structure below A is irrelevant.
```

{ } = indication of parallel choice.

```
    ( ) = optional item

X - YZ  = dash indicates boundary of that
1   2     term. Thus term1 = X, term2 = YZ.
```

APPENDIX A

TRANSFORMATIONAL GRAMMAR

The theory of transformational generative grammar adapted from [WI83], as developed by Chomsky [CH57] has come to dominate theoretical linguistics, and its adherents have dealt with many issues that bear on our understanding of language as a cognitive process. There are three distinct but related things at times labeled "transformational grammar."

(1) There is the general approach to linguistic theory--the generative paradigm. Here, the problem of linguistics is seen as that of understanding syntactic competence--the formal structure that underlies the intuitions of a native speaker about the grammatical structure of sentences. While it emphasizes the formal characterization of grammaticality, and its research methodology is dominated by the development of syntactic formalisms that provide precise means for characterizing the set of sentences of a language; it is far from being a coherent, consistent paradigm with an agreed upon set of questions and methodologies at all levels of detail [WI83].

(2) There is a particular model of language within the generative paradigm that might be called the transformation model. It has served as the basis for much of the work in generative linguistics. It is however not the only possible

generative model, and much of the linguistic research that is described as "non- (or anti-)transformational" argues against this model while maintaining the same generative approach to language and grammar (e.g. context-free grammar).

(3) Within the transformational model, there are several distinct formal mechanisms. Among these, exists one particular kind called the transformation rule.

Approaches to Structure

A formal linguist describes the structures of a language by devising a collection of rules, called grammar, that can be used in a systematic way to generate the sentences of a language. Our intuitions about language call for some kind of constituent structure [WI83]. Sentences are made up of "phrases" or "chunks" and this structuring is important for how they communicate meaning.

Dependency Tree (head and modifier) Approach

Many traditional representations of grammar describe structures in terms of a basic simple pattern that is elaborated, or modified. The underlying structure of a sentence for example can be thought of as a noun followed by a verb, with additional words (such as articles and adjectives) modifying the noun and others (such as adverbs) modifying the verb. Modifiers can themselves be modified (by words such as adverbs) to produce a multi-layered structure. At each level, there is a node, the contents of which is a word, called the head. The children of the node are its modifiers.

The Phrase Structure Tree(immediate constituent) Approach

Immediate constituent grammars allow any constituent itself to be a constituent structure. Each node has a sequence of siblings and has as its constituents a label which corresponds to a word or phrase (syntactic category). Leaf nodes have words as their labels and all other nodes have syntactic categories.

The Slot and Filler (Role Structure) approach

The motivation for analyzing the structure of a sequence of words is that the result is to be used in some further analysis or processing, such as translation or comprehension of meaning. The fact that there is a particular sequence of recognizable elements is one aspect of structure, but this does not capture a functional relationship. The pattern S = NP VERB NP, for example, does not specify that the two instances of NP play different roles--one is the subject, and the other is the direct object. In this structure the pattern for each kind of phrase is described as a sequence of named "slots" each of which corresponds to a "filler" in the sequence to which it is matched. As with the constituent structure approach, each pattern element specifies the kind of word or phrase it can match, also the name of the role that the word or phrase plays in the pattern as a whole. A role name can appear only once, but the objects filling different roles may be of the same kind. In addition, it has a role (register) table which associates role names with other nodes, sequences of nodes, or symbols. The keys in the role table are slots and the corresponding entries its filler.

The Inadequacies of Phrase Structure Grammars

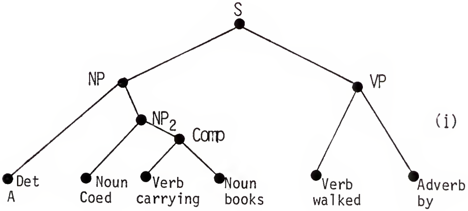
Transformational Grammar developed in the context of structural linguistics. Until the development of formal grammars the discussion was couched primarily in terms of immediate constituent structuring.

In describing a language, a linguist produced a set of rules such that an immediate constituent structure conforming to those rules could be assigned to every sentence of the language. The rules specified the composition of constituents (siblings) allowed at a node of a given category. There were, however, two basic assumptions that led to problems (1) tree-like structuring and (2) the locality of rules--each rule dealt with a single node.

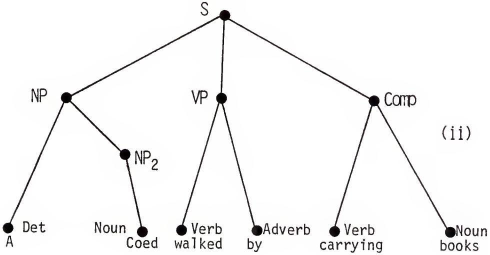
In assigning structures to discontinuous constituents consider:

A coed carrying books walked by [1]
A coed walked by carrying books [2]
She called up her poor lonely aunt last week [3]
She called her up long distance once a week [4]

Naturally, for [1] we expect a structure like that of tree(i) (Figure A-1) in which the phrase "A coed carrying books" is a single constituent. Since there is no way to attach "carrying books" to "A coed" without violating the ordered tree-like nature of the constituent structure or giving up the notation of a sentence as NP followed by a VP we are forced to assign either (ii) or (iii) to [2].



(i)



(ii)

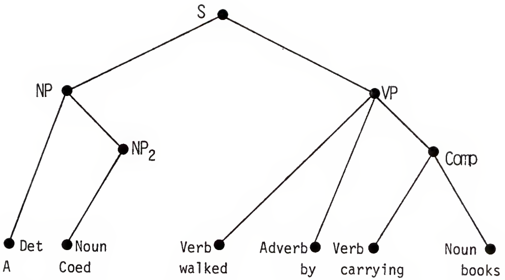


Figure A-1. Immediate Constituent Structures for Related Sentences.

While it is possible to write a grammar that generates all three trees by including rules that allow modifiers like "carrying books" to appear in the relevant places, it is awkward and results in building structures that do not correspond to our intuitions about meaning. Our understanding is such that "carrying books" is a modifier of the coed regardless of where it appears and consequently we would like the structural description to reflect the connection.

Similarly, in [3] and [4] if we simply treat "call up" as two unrelated words, the grammar fails to reflect the underlying structure in which they are really a single unit. We would thus like to treat both as making use of a single lexical item(or idiom) "call up." In a tree structure for [4] they cannot form a constituent by themselves since "her" is interposed.

The matching of attributes between a verb and one or more of the major nouns in a sentence or between a noun and its modifying adjectives is a common phenomenon in most languages. English is relatively deficient in this area but does demand an agreement in number between its subject and verb. Thus distinctions (singular, plural) are very critical to analyzing the meaning of sentences. Agreement does not just apply between adjacent constituents but can also involve a gap. There is, however, no way to reflect this agreement in a tree structure that would match our intuitions about constituent structure.

These problems were recognized and solutions proposed with varying degrees of complexity, and it was assumed that through some amount of patching, the basic ideas of constituent structure could be extended. A major shift occurred after the publication of Chomsky [CH57]. He argued that an entire dimension of linguistic structuring was being left out, and that the theories he proposed would illuminate this additional dimension as well as solve many of the detailed technical problems of constituency theory.

The phrase structure analysis that is directly associated with a sentence is superficial in that it is determined by the ordering of elements in the sentence. A native speaker has intuitions about similarity of sentences that are not based solely on ordering but are still based on syntactic structure rather than on meaning. With sentences which are paraphrases of each other, the connection depends on some intuition of the underlying similarity of structure.

Chomsky's proposition was much more than a simple refinement of ideas of constituency, since it meant dealing with structures that could not be observed in the data but were postulated to have some kind of underlying mental reality.

Generalized Rewrite Rules

A grammar rule, is a formal object specifying a possible structural relationship among phrases and words in a sentence of a language. It specifies the legal steps in a process of modifying a sequence of category symbols. A formal definition embodies assumptions about the nature of the structures. In the case of assumption of tree-structured constituency and the locality of rules, the relation of grammar rules to linguistic structure can be described as [WI83]:

A sequence of words is in the language defined by a grammar if there exists a phrase structure tree such that:

1. The leaves of the tree correspond in order to the words of the sequence.
2. Every node whose label is a lexical category has a single child which has as its label a word belonging to that category.
3. Every other node in the tree corresponds to one rule of the grammar, with the label of the node being the left-hand side of the rule and the sequence of labels of its children matching the right-hand side.

We shall now define a grammar more formally as a 4-tuple

$$G = (V_N, V_T, S, P)$$

where V_N, V_T are disjoint sets of non-terminal and terminal symbols.

V_N = { Variable(symbol) representing a syntactic class within the grammar and are those that appear in the left-hand side of any rule; for English, it represents the larger syntactic units such as Noun Phrase, Verb Phrase, Prepositional Phrase, and Sentence}

V_T = {Those symbols that do not appear in the left-hand side of any rule; for English its usually the set of syntactic parts-of-speech}

S = {Commonly called the goal or starting symbol is the distinguished symbol of V_N--a nonterminal symbol that appears as the sole member of the left-hand side of one or more rules}

P = {A finite set of productions}

The set $V = V_N \cup V_T$ is called the vocabulary of the grammar.

This description fails to address how a sentence is analyzed or generated. It treats grammar rules as constraints on phrase structure trees and it is a static characterization of what constitutes a valid analysis--a tree structure that describes the sentence in terms of constituents based on grammar.

The deficiencies of phrase structure grammars are really limitations on the usefulness of this kind of constraint. The solution proposed by Chomsky [CH57] for the problems of immediate constituent grammars grew out of his attempts to formalize the kind of analysis such grammars provided. Instead of treating grammar rules as constraints on the structures assigned to a sentence, he noticed that they could be treated as rules for specifying steps in a derivation. A derivation is simply a process by which an analysis meeting the constraints is generated step by step. The derivation does not add information to the phrase structure analysis, since the final tree structure uniquely specifies which rules were applied, and the order in which the nodes are expanded in the course of the derivation has no effect on the structures that are produced.

A more general notion of derivation and rewrite rule permits structures to be manipulated in the course of a derivation without producing corresponding structures in the final result. The simplest generalization of rewrite rules does not involve trees at all, but is based on string derivation as defined in Figure A-2.

It allows the generation of sentences of the language without keeping a structural description. The working structure is a sequence of symbols, which includes both symbols of the grammar and words. As each rule is applied, one of the symbols is replaced with another symbol or sequence of symbols, eventually leading to a sequence containing only words. At each step of the rule, one symbol of the sequence is replaced with a sequence corresponding to the right-hand side of some grammar rule. Figure A-3 illustrates a grammar based on a more general kind of rewrite rule.

```
Purpose: Produce a sentence using a C-F grammar without
         deriving a tree structure
Background: a C-F grammar and a dictionary
Results: a sequence of words that is a sentence in the
         language
Working Structures:
  Derived sequence: a sequence, each member of which is
         either a syntactic category or a word; initially
         consisting of one occurrence of the distinguished
         symbol of the grammar
Method: repeat:
      - If the derived sequence contain only words, return
         it as the result.
      - choose any position in the derived sequence that
        contains a syntactic category
      - if the category is:
          - a lexical category, then choose any word in the
             dictionary belonging to that category, and
             replace the symbol at that position with the
             word.
          - a composite category, then choose any rule in
             the grammar having the symbol as its left-hand
             side, and replace the derived sequence with a
             new sequence that consists of:
             - the part of the sequence that was before the
                chosen position
             - the sequence of symbols specified the right
                hand side of the rule
             - the part of the sequence that followed the
                chosen postion
```

Figure A-2. Schema(nondeterministic) to derive a sentence without a tree structure.

```
1   S             -->  NP VP
2  NP             -->  Determiner Noun Number
3  Number         -->  Singular
4  Number         -->  Plural
5  Singular VP    -->  Verb -s
6  Plural VP      -->  Plural Verb or VP --> VERB[PLURAL __]
7  Plural         -->  -s
```

Grammatical markers: Singular, Plural

Figure A-3. A generalized rewrite grammar

This kind of grammar is not context-free(C-F). It differs in three ways. (1) It allows a sequence of symbols rather than a single symbol, on the left-hand side. (2) In addition to syntactic categories and words, it allows special grammatical markers. (3) It allows the right-hand side of a rule to be empty. At each step of the derivation, a rule is chosen whose left-hand side matches some sequence of adjacent symbols. That sequence is removed and replaced with the right-hand side of the rule. This kind of grammar can be used to indicate context dependencies by putting more than one element on the left-hand side of a rule. For example, rule 6 is a context-sensitive(C-S) rule. This is also written with the left-hand side consisting of a single symbol that is then expanded, with other symbols appearing in brackets as a context following the rule. The underline indicates that the VP must appear following the marker PLURAL in order for this rule to apply.

A grammar that generates same sentences could easily be written using only C-S rules. However, there exist languages that can be generated by a general rewrite grammar but not by one containing only C-S rules.

In extending the notion of rewite rule beyond C-F grammars, it is possible to provide mechanisms that build underlying structures that are intuitively more appropriate. It is thus possible to allow a combined constituent like "call up" to be generated as a unit and then use rewrite rules that interpose pronouns to produce a sentence "He called her up." This implies that we can extend the class of languages for which we could write a grammar; for Chomsky [CH57] developed a formal theory of grammars to offer such an extension. He devised a classification of grammars into a hierarchy according to their power. There are four types of grammars (Figure A-4), each defined by the kind of rewrite rules it contains in descending order of their generative power, and in ascending order of their simplicity.

Type	Name	Restriction on rule form	
0	Unrestricted	None	$\Theta \ \varepsilon \ \{V_N \ \mathrm{U} \ V_T\}^*$; $\Phi \ \varepsilon \ \{V_N \ \mathrm{U} \ V_T\}^*$
1	Context-Sensitive(C-S)	The right-hand side consist of the left-hand symbol with a single symbol expanded	$\Theta = \alpha A \beta$; $\Phi = \alpha w \beta$ where $\alpha, w, \beta \ \varepsilon \ \{V_N \ \mathrm{U} \ V_T\}^*$ $A \ \varepsilon \ \{V_N\}$ $A, w \neq \{\Omega\}$
2	Context-Free(C-F)	The left-hand side consist of a single symbol	$\Theta \ \varepsilon \ \{V_N\}$ $\Phi \ \varepsilon \ \{V_N \ \mathrm{U} \ V_T\}^*$
3	Regular or Finite State (FS)	The right-hand side consist of a single terminal symbol or a terminal symbol followed by a single non-terminal symbol	$\Theta \ \varepsilon \ \{V_N\}$ $\Phi = V_N V_T$ or V_T

* any string except the null string $\{\Omega\}$

Figure A-4. The hierarchy of grammar types

Type 0 is the most powerful, and each increasing number is more restricted. For each grammar type, there is a class of languages that can be defined only by a grammar of that power or greater power. The entries for rule restriction are cumulative--the rules of a type 3 grammar obey the restriction for types 1 and 2.

Pattern or network grammars are unable to generate many simple languages (a language consisting of the sequence say $A^n B^n$). This language can easily be defined by a type 2 grammar with two rules: S --> ASB and S --> AB. Finite state grammars as defined above (and Chapter 3) are the easiest to implement using the simple finite state sequential machines of automata theory. Unfortunately, it may be shown that there is a large class of English sentences (those with center embedding, and relative clauses) which cannot be described with FS grammar. In other instances, natural analysis may be the recursive one in which each constituent is part of one starting to the left of it. Many of the embedding phenomena of natural language can be changed to repetitive structures at the cost of losing the natural structuring of the meanings. The addition of recursion (the ability of a non-terminal symbol to replace itself) allows recursive transition networks to possess the generative power of Context-free (C-F) grammars. These grammars are able to describe a wider class of English sentences but a more powerful formalism, attributed to Woods [WO70] is the Augmented Transition Network (ATN), see Chapter 3.

Going up in hierarchy, a construction that cannot be handled properly by C-F grammar is the use of "respectively" in English (i.e. dependencies that can be characterized as an instance of the sequence:

$$x_1\, x_2\, x_3 \ldots\ldots\ldots\ldots y_1\, y_2\, y_3$$

$$x_i R\, y_i \qquad\qquad \forall\, i \quad)$$

Chomsky[CH57] showed that English is neither a C-F nor a regular grammar. It is difficult to find examples in natural languages of structures that could not in some way be generated by grammars low in hierarchy. The more relevant problem is that the natural intuitive analyses require the building of structures that cannot be the result of an analysis produced by a simple grammar. As an example, the problem of subject-verb agreement can be handled in a C-F grammar by having parallel expansion as in the grammar:

$$\begin{array}{lll}
S & \rightarrow & S_{\text{singular,plural}} \\
S_{\text{singular,plural}} & \rightarrow & NP_{\text{singular,plural}} \; VP_{\text{singular,plural}} \\
NP_{\text{singular}} & \rightarrow & \text{Determiner}_{\text{singular}} \cdots \\
\cdots & & \cdots
\end{array}$$

Although this method makes it possible to define the language with C-F grammar, it means that the distinction between singular and plural must be propagated through all of the constituents involved, duplicating all of the rules. If other distinctions are added (as in person or gender agreement), the effect would be to multiply the size of the entire grammar by the number of terms in each such new distinction. This could result in a combinatorial explosion!

Thus the motivations for using more powerful grammars go beyond "weak generative capacity" (the ability to specify a given language) but are based on a desire to have a grammar that is simple and that produces structures that correspond to our intuitions about other considerations such as meaning.

If we view a grammar as controlling the derivation process rather than as specifying the derived structures, we can offer a solution to the problems of similarity and difference. For if two sentences have underlying similarities that are not apparent in their structure, we can postulate that at some earlier point in the derivation the similarity was explicit, and the application of subsequent rules caused the difference in appearance. Conversely, two sentences that look alike might have differed in significant ways earlier in the derivation. We can thus create grammars that perform derivations on structures. Instead of having rewrite rules that operate on sequences, we can have them operate on tree structures. One of the greatest appeals of transformational grammar has been its ability to represent such derivations succinctly.

The Basic Transformational Model

The generalized rewrite rules provide the formal capability to characterize a wide variety of languages. In the development of transformational grammar, it has been argued that by understanding the kinds of grammar that are best suited for human languages we will gain a key to understanding the structure of the human mind. Here we present a simplified model of the "standard theory" [CH65]--sometimes referred to as the "Aspects model."

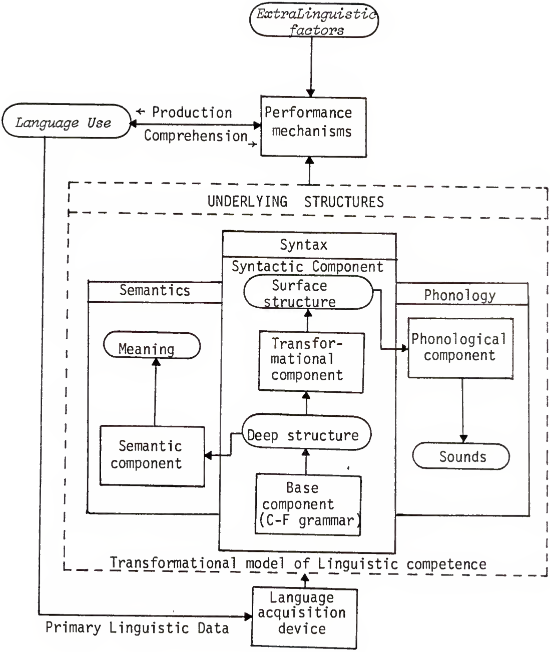


Figure A-5. Model of the human language capacity.

Transformational Model of Linguistic Competence and the Human Language Capacity

Figure A-5 illustrates the basic generative model of the human language capacity. The model has three main components with the central being the linguistic competence of the adult language user. Linguistic competence consists of three basic components: phonological, semantic, and syntactic. The three together make up a person's knowledge of his language. In the standard transformational grammar approach, it is assumed that these three components can be characterized separately. This characterization called the "autonomy of syntax hypothesis" has been mistakenly caricatured at times as a belief that meaning is unimportant in the study of language. Adherence to the autonomy principle has led to an atmosphere in which the study of syntax has been given priority, but in theory it is simply a methodological assumption about the degree of relatedness between meaning and syntax.

The claim is essentially that an analysis of the structure of a language which includes sound patterns and meaning will be best achieved by finding the structure of each component separately and then understanding their interactions. The idea of treating a complex system as nearly decomposable does not depend on total independence but rather that the interactions among components are much less crucial than the independent functioning within each. We can define syntactic structure and grammaticality purely in syntactic terms, meaningfulness in semantic terms, and sound patterning in phonological terms. An overall picture is then achieved by describing correspondences between them.

As was mentioned earlier, the simple ideas of phrase-structure grammar do not suffice for dealing with actual languages and languages in which same sentences with differing visible structures seem closely related while others with same apparent structure seem essentially different. What then is the nature of syntactic structure of a language? The solution was to posit (postulate) two levels of structure--deep and surface structures--which are related by grammar.

In the standard transformational model, the base component is a context-free grammar that generates the deep structures which contains everything relevant to its meaning. The deep structure can undergo a variety of "transformation" of form (word order, endings, etc.) on its way up while retaining its essential meaning [BA81]. The transformational component consists of a set of rewrite rules (transformational rules) that operate on tree structures called phrase (P-) markers (these rules map one p-marker onto another p-marker; that is, they change the structure of a tree in some way to produce another "derived" tree). This component is used in a derivation process that begins with a deep structure produced by the base and results in a surface structure, which can then be used to produce the actual sequence of sounds or letters in the sentence. A sequence of words is a sentence in the language defined by a transformational grammar (TG) if it can be generated by a derivation that produces a deep structure and then transforms it into a surface structure that has a sequence of leaf nodes corresponding to the sequence of words.

The linguistic competence, along with extralinguistic factors, is the basis on which the performance mechanism carries out linguistic activities such as comprehension and production of sentences. It is also the source of our intuitions about grammatical structures and can therefore be studied with research methodology based primarily on "grammaticality judgements" by adult native speakers. The language acquisition device (LAD) is the means by which a person comes to know a language. Its input is the "primary linguistic data" of utterances heard by the language learner and its output is the collection of cognitive structures that embody the knowledge of a particular language. These structures include the grammar as well as rules that deal with sound patterns and meaning.

One of the major ways in which the transformational model (TM) differs from most work in the computational paradigm is the assumption that the underlying competence can be described independently of the processes of comprehension and production. C-F grammars offer a good example of such separability. For we can formulate a C-F grammar for a language and understand it as a set of constraints on valid analyses of sentences without introducing questions of how these sentences are understood or produced. We can build parsers and generators that make use of the grammar, keeping details of the algorithms separate from the basic characterization of the language. As one obvious effect of separating the underlying structure from the processing, we can use the same grammar, with different procedures, for both generation and parsing.

In the TM, it is assumed that this property holds for the more complex syntactic formalisms of TG. The model distinguishes competence(via structures), from performance(the process that makes use of them). To justify this separation, Chomsky [CH65] argued that the details of how language is actually spoken or understood is a function of such phenomena as memory limitations, changes of intention in the middle of utterances, and even physical states such as coughing or sleepiness, which accordingly are not properly part of linguistic theory.

In many subsequent discussions, all issues of processing were dismissed as being too intertwined with these nonlinguistic issues and not at the appropriate level of idealization for linguistic theory.

The TM has been misinterpreted at times as a computational model of how language is produced, since the process of derivation leads to the generation of a sentence. To the transformational linguist the model is clearly something else. It is a mathematical abstraction, intended to characterize formally the knowledge a person must have about the syntax of a language. There is no assumption that a person generates/parses sentences in a way that corresponds to the derivation process.

Chomsky's [CH57] proposition for grammars was for a three-fold organization: (1) a phrase-structure grammar generating strings of morphemes representing simple declarative, active sentences, each with an associated p-marker or derivation tree, (2) a sequence of transformational rules (TR) rearranging the strings and adding or deleting morphemes to form representations of the full variety of sentences, (3) a sequence of morphophonemic rules would map each sentence representation to a string of phonemes. Later work has since changed this model of the grammar [BA81]. Suppose the phrase-structure grammar is used to to produce the derivation tree illustrated in Figure A-7a. To generate "The giraffe ate the apples" one would apply transformation mapping

TENSE + eat --> eat + PAST

A morphophonemic rule then maps

eat + PAST --> ate

Similarly, "The giraffe eats the apples" is obtained by using the TR to select present tense and, because the verb follows a singular NP, would map

TENSE + eat --> eat + s

The TR must look at nonterminal nodes in the derivation tree to determine that "the giraffe" is in fact NP_singular. This illustrates the broadness of the TR as opposed to phrase-structure grammar.

The Form of Transformation

The problem of precise formal specification of transformation rules has been a continuing theme of disagreement and research in TG. There is no standard accepted notation for transformation rules; the following is a version adapted from Akmajian and Heny [AK75]. The rules include many features such as the use of the symbols "+", ">". "<", and "#". The "+" as in 1+4 (Figure A-6) indicates sibling adjunction: the deleted category (e.g. NP) is adjoined as a sibling of the verb. The ">" indicates that the node for "be" is to be adjoined as the rightmost child of the node to its left in the sequence, while "<" indicates adjunction as the leftmost child of the node to its right.The "#" indicates "Chomsky adjunction," in which the new node and the one already there are joined as children of a new parent node whose label is identical to the label of the previous node.

A transformation rule (TR) consists of three components: (1) a structural description (SD) which is a pattern to be matched against the tree as derived so far. This SD plus a set of conditions on the tree that is matched together correspond to the left hand side of a generalized rewrite rule, since they control whether the rule is applicable. The SD need not specify an entire tree but only a sequence within it. A necessary and sufficient condition for a set of nodes to match is that 1)they all be neighbors(no interpositions allowed), 2) a structural change (SC) which is an operation to be done on the tree structure when the rule is applied i.e. once it has been determined to match the SD. The SC is the right-hand side, 3) a set of conditions on the tree that is matched.

```
------------------------------------------------------------
PASSIVE(Optional)

SD:  NP           Aux            Verb          NP
      1            2              3            4

SC:  4            2 > be+ -en    3             by # 1

  :follows DATIVE, precedes THERE-INSERTION, TAG FORMATION ...
------------------------------------------------------------
DATIVE(Optional)

SD:  Verb          NP            { to  }       NP
                                 { for }
      1            2               3            4

SC:  1 + 4         2               0           0

  :precedes PASSIVE
------------------------------------------------------------
AFFIX HOPPING(Obligatory)

SD:     { Pres   }          ( { modal } )
        { Past   }          ( { verb  } )
        { -ing   }
        { -en    }
           1                    2

SC:        0                  2 # 1

  :precedes DO-SUPPORT, follows SUBJECT-VERB INVERTION
------------------------------------------------------------
```

AGENT DELETION(Optional)

SD:	NP	X	be en	V	Y	{ by someone } { by something }
	1	2	3	4	5	6
SC:	1	2	3	4	5	0

:follows PASSIVE, preceds Modifier shift, unordered w.r.t. Affix Hopping

---

EQUI-P DELETION(Obligatory)

SD:	X	(NP)	Y	[{Comp} [{Poss} S[{for }	NP	Z]]]S	W	(NP)	R
	1	2	3	4	5	6	7	8	9
SC:	1	2	3	4	0	6	7	8	9

where 2 = 5 or 5 = 8
2 and 8 may not both be null
with main verbs such as "want", term 1 must be the subject of the verb; with verbs such as "force," term 1 must be the object of the verb.

Figure A-6. Some transformation rules

The notation makes it possible to specify transformations precisely. However Bach [BA74] notes:

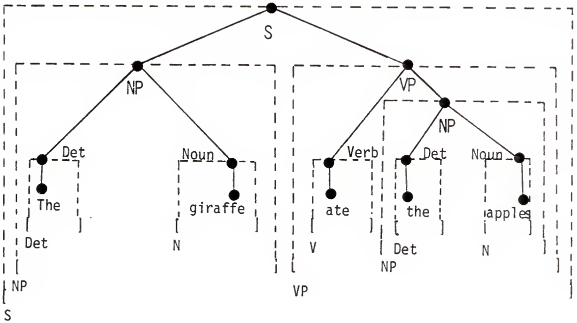
> The reader should be warned that many discussions of transformation for various languages are stated in a very informal and rough way, so that the exact effect of a given transformation is often unclear. (pp.88)

In addition to specifying elements by syntactic category or by giving individual words or markers, the SD can include alternatives, indicated by a set of items in curly brackets. This element matches if any one of the alternatives matches. The parentheses in the SD indicate that this element may or may not be present. A transformation is not seen as taking a sequence of elements and scrambling them, but rather as performing a series of elementary(e-) transformations in the tree structure. Each e-transform is a deletion (indicated by null '0' symbol at the corresponding place in the sequence), a substitution, or an adjunction. The different adjunction symbols control the way in which the new element is attached.

Generally, arguments about the formulation of transformations do not hinge on the finer points of the details of the produced structures and can go on without a precise specification. In instances where it is necessary to be precise as in a computer implementation of TG (e.g. [FR71]), it has turned out to be quite difficult to do this consistently.

Some transformations (Figure A-6), such as PASSIVE, are optional. Their application in the course of deriving a sentence results in a particular form; non-application yields a different but equally grammatically correct one. Other transformation marked "obligatory" ensures agreement in number of the subject and the verb. They perform "housekeeping" activities that must be done to get things into their appropriate final form, and if they are not applied, the result is not a grammatical sentence.

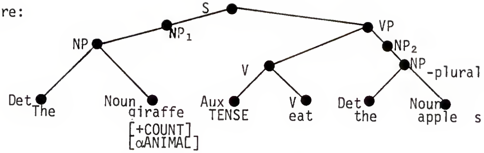
A good example of an obligatory transformation is AFFIX-Hopping, which can be applied more than once in a single sentence.



$$\left[_{S}\left[_{NP}[_{Det}\ \text{The}\ ]\quad[_{N}\ \text{giraffe}]\right]\left[_{VP}[_{V}\ \text{ate}\ ]\quad\left[_{NP}[_{Det}\ \text{the}\ ]\quad[_{N}\ \text{apples}]\right]\right]\right]$$

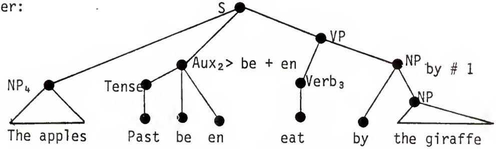
Figure A-7a. Linear Equivalent of the phrase marker.

before:



SD:	NP_1	Aux	V	NP_2
	1	2	3	4
SC:	4	2 > be + en	3	by # 1

after:



Passive (Optional) "The giraffe ate the apples"

Figure A-7b. Structure changes specified in the Passive transformation.
Figure A-7. Passive transformation.

For a number of reasons, one is led to have the base component generate the elements of the verbal sequence in an order that is different from the order of the appearance in the surface structure. Transformations like PASSIVE maintain the base ordering. To obtain "The apples were eaten by the giraffe" it would be necessary first to apply the optional PASSIVE transformation (Figure A-7) changing the string analyzed as

NP_1 + Aux + V + NP_2

--->

NP_2 + (Aux + be) + (en + V) + by + NP_1

Thus the optional changes "The giraffe TENSE eat the apples" --> to "The apples TENSE be(en eat) by the giraffe" then forces agreement of the auxiliary verb with the new plural subject. Further application of the obligatory rule yields

"The apples be PAST eaten by the giraffe"

where

"be + PAST" as opposed to "be + s + PAST"

is ultimately mapped to "were."

Optional and Obligatory Transformation

In early versions of TG there was a simple distinction between optional and obligatory rules. Housekeeping rules were obligatory (EQUI-NP DELETION, AFFIX HOPPING) and others, such as NEGATIVE, PASSIVE, QUESTION FORMATION, DATIVE, AGENT DELETION, were optional. Deep structure, interpreted by the semantic component, determines sentence meaning. It follows that the application of all transformations were meaning preserving.

AFFIX-HOPPING is an obligatory transformation and can be applied more than once in a sentence. The sentence

"Shean has been running" is based on the sequence

PAST have -en be -ing run

This rule applied three times results in

have - PAST be-en run-ing

where the constituents have been reordered and grouped because of the "#" in SC indicating adjuction. This is then converted by the phonological component to "had been running."

In Fig.A-7, the SC of the PASSIVE rule insert be-en before the verb eat. AFFIX-HOPPING reverses the order of -en and eat and combines them into a single constituent. The phonological component converts the sequence "eat - en" into the participle form eaten.

To account for these facts, the PS rule that expands auxiliaries is modified to include the affixes -en, -ing:

Aux --> (Modal) (have en) (be ing) ....(1)

Rule: Every affix that is immediately followed by a verb must be placed on that verb, becoming part of it.

Thus, every affix must be attached to the verb immediately to its right, if a verb is there.

To account for the distribution of past or present tense forms in the first verb (1) is modified to

Aux --> Tense (Modal) (have en) (be ing)
Tense --> {pres}
{past} ....(2)

where we have introduced either "past" or "pres" as the first obligatory member of Aux. As long as Tense is considered to be an affix, AFFIX HOPPING will attach past/present--whichever occurs in a particular sentence--to whatever verb is generated immediately to its right (the first overt verb in the string). The rule ensures that the verb following a modal--a subclass of the auxiliaries--will have no affix. The Affix Hopping transformation rule guarantees (1) that the first verb in any series of verbs will have the correct tense attached to it; (2) that the verb following a modal will have no affix; (3) that the verb following "have" will have -en, the perfective ending and that the verb following "be" will have -ing.

The Agent Deletion rule is an example of an optional rule and has the effect of deleting the "agent phrase" (by someone/by something), in a PASSIVE tree. For instance, beginning with a deep structure corresponding to the active form "Someone robbed the man," we can apply (see Appendix B-4) PASSIVE to obtain "The man was robbed by someone," and then Agent Deletion to derive "The man was robbed." Here, the "someone" is not an actual lexical item but a special pro-form used to stand in for NP. Thus when deleted, no information is lost.

EQUI-NP DELETION (EQUI) is an optional rule which deletes the subjects of embedded sentences under two conditions[BU71]:

1. The verb of the directly higher sentence must belong either to the class "expect," "want," etc. or to the class of "force," "bribe," etc.
2. The subject NP of the embedded sentence must equal either the subject NP of the matrix sentence, or the object NP of the matrix sentence.

The lower(embedded) sentence along with the complementizers seem to have all the components of a sentence: NP - Aux- VP.

When the complementizer is "that," the complement sentence itself is in the same form it would take if it were an independent sentence standing alone.

One way of formalizing the YES-NO Question rule may be stated roughly as

SD:	NP	Tense	({modal}) ({have }) ({ be })
	1	2	3
SC:	2 + 3	1	0

Figure A-8. Yes/No Question formation (Optional)

Here, the element to be moved is the tense, along with the following modal, "have" or "be" if there is one. Otherwise, Tense will ultimately have the auxiliary verb "do" attached to it via another obligatory rule known as DO SUPPORT, to form a question. Subsequent versions of TG adopted a convention that all transformations were meaning preserving. If the deep structure captures all the meaning of a sentence, then a transformation cannot turn a statement into a question. Instead, markers were generated in the base structure and the corresponding transformations were made obligatory. In the standard model, the grammatical marker Q(questions) is generated in the base structures, and there is an obligatory transformation:

SD:	Q	NP	Tense	({modal}) ({have }) ({ be })
	1	2	3	4
SC:	1	3 + 4	2	0

Figure A-9. Yes/No Question formation (Obligatory)

Other transformations later delete the marker Q before the surface structure is reached. Similarly, other meaning-changing transformations such as those for negation and the formation of imperatives can be triggered by appropriate markers. Those that do not change meaning, e.g. Passive, are unaffected. Those also can have effect on meaning, and hence led to further modifications to the framework.

Revisions to the Model

One of the strong appeals of early transformational grammar was that it would be used by linguists to predict syntactic intuitions about a large number of complex structures with a relatively simple formalism and a limited number of rules. As more linguists applied it to a variety of problems, it became apparent that many facts about English demanded further additions to the theory, along with specialized rules, e.g. the transformational cycle covered a great deal of territory, but other phenomena did not yield as easily. There were a number of circumstances under which transformations that seemed quite general could not apply. Further rules and constraints had to be added. Each of these in turn faced the problem of having exceptions if stated in a general way and being inordinately irregular if it tried to account for all of the cases.

Basic Restrictions

A principle in transformational theory is that all of the rules needed to describe a natural language are structure dependent--the arrangements can be stated in terms of phrase structure trees, not calling for arbitrary rearrangement of symbol strings. The limitation that transformations be structure dependent is enforced by the nature of the SD--by the fact that it can only specify an analysis in terms of tree nodes. For some transformations, it is necessary to provide further limitations on the structures to be matched as in EQUI (Fig. A-6) which produces structures from a deep structure having an embedded sentence. The subject of the lower sentence is deleted, since it is identical to the subject of the matrix sentence as indicated by the condition 2=5 (a specific instance of a more general restriction called recoverability of deleted structures). In addition to the conditions given explicitly, there is a notation illustrated by labeled square brackets. Instead of just specifying a sequence of elements, the SD also includes a further indication of the tree structure. It requires that in addition to being neighbors, these elements must be descendants of some node labeled S. There is no standard agreement on the exact form that the structure specification can take within SD's. There is an element X standing for any intervening set of structures. It need not be matched against a single tree element but can cover any number of nodes. Hitherto, each element of the SD specified particular syntactic categories, markers, or words. At issue is the question of the appearance of "essential variable" in transformations; for example, to handle a situation like the intrusion of an adverb in a sentence. However, this can be handled without variables either by allowing an optional ADVERB element(as with the MODAL element of Fig.A-6), or the grammar can be written so that adverbs are generated elsewhere (e.g. at the end of the sentence) and moved into position after transformation like Passive has been applied. Also, there are implicit variables at the beginning and end of every SD since it need not match the entire sentence but only a contiguous segment.

A transformation can delete a phrase only if there is sufficient redundancy in the structure that no information is lost. This redundancy may be due to the appearance of another copy (as in EQUI); the deleted item is a word or grammatical marker (as with "to," "for" in DATIVE); or the deleted item is a dummy, or pro-form, that is inserted in the base structure to fill a node which is later deleted (AGENT DELETION).

There has been an argument that through this and related restrictions, transformational grammars could express all of the generality of natural language with mechanisms formally less powerful than unrestricted type 0 grammars, and that this would represent an empirical fact about the human mental apparatus.

Different formulations of transformations has been devised with this in mind but generally, the basic restrictions of recoverability has been kept. So far, only EQUI transformation deletes a component other than specified words or grammatical marker without adjoining or substituting it elsewhere in the structure, with its condition guaranteeing no loss of information. There are other transformations, however, which can lead to losses of information.

Copying is used in cases when possessive pronoun must be a copy of the subject NP of the sentence, as in

NP Aux crane neck

---->

NP Aux crane [NP's] neck
[+pro]

Figure A-10. Subject copying

The feature [+pro] suggests that the possessive modifier of "neck" will be a pronoun (John craned his neck).

Subject copying rule applies to the string:

you crane ____ neck

---> subject copying

yielding

you crane your neck ....(1)

There are many other cases in syntax where exactly this kind of rule (reflexive) is helpful to operate on certain deep structures and convert them to the proper surface structures. In sentences that appear not to have subjects as in "Come !" "you" is understood to be the subject. We might say the subject "you" is actually present in the deep structure, and that there is a rule which deletes it. There is evidence for the existence of such a rule--the reflexive rule. The fact (1) that imperative sentences are understood to have "you" as their subject, together with (2) only reflective pronouns occur in imperatives,suggests that there is a rule which deletes "you" under some conditions, after the reflexive rule has applied. This rule is called imperative:

Imperative

you - Aux - VP

--->

0 - 0 - VP

Applied to (1)--the output of the subject copying rule (Appendix B-2) explains the fact that we have

you crane your neck

---> imperative

crane your neck ....(2)

It is this kind of explanation which transformational grammar seeks to give for sentences which are or are not acceptable.

Addition of Syntactic Features

In early versions of the grammar, phenomena like agreement were handled by introducing grammatical markers such as PLURAL, FIRST-PERSON, MASCULINE, and moving them around with transformation. In the standard theory, many of the phenomena are treated by assuming that a tree node whose label is a lexical category can contain a set of syntactic features in addition--as in the REFLEXIVE transformation (Appendix B-3). The feature +REFLEXIVE is appended to the node, eventually leading to its conversion--through other obligatory transformations--to the pronoun "yourself." Features are indicated in square brackets underneath an element of the SD or SC. If they are in the SD, the corresponding node in the match must already have that feature for the rule to apply and it will remain there afterwards. If features are in the SC, they are added to the node when the change is applied. In addition to agreement features for person, number, and gender, features are used for other things. Nouns for example are classified as +/- COUNT. A +COUNT noun like "apple" cannot appear in the singular without a determiner, while a mass noun (-COUNT) like "rice" can. Using features, a rule for number agreement simply needs to copy the appropriate features from the place they are generated--here in the NP--to the other places they are needed. When features are copied they are also left on the original node. Greek alphabets e.g. α , β (Fig.A-7) are used in place of +, - to indicate that whatever value of the feature appears in the SD should be copied to the SC.

Extensions to the Base Grammar

The underlying phrase structure trees used in the transformational grammar are not identical to those generated by the context-free grammars. In order to simplify the transformational component, a number of extensions were made to the base grammar, although it is still formally C-F grammar.

Features. The use of features in transformations implies that the base component must generate nodes containing features that do not correspond to words of the language, as well as labels--categories, words, and markers. Examples are the 'Q' marker used to trigger question formation and the dummy pro-form used for the subject of sentences that will be passivized and have their agent deleted. Formally, a set of features can be thought of as a complex symbol and does not increase the power of the grammar. If this convenience is used freely, it makes it possible to write rules that perform whatever computations could be stated for any given computer. If they are not used at all, it is difficult to write a sensible grammar.

Embedding. In early versions of the grammar, the distinguished symbol S, never appeared on the right-hand side of a rule, although the formalism does not preclude that possibility. Embedding and conjoining were handled with the aid of variables(X). The separate sentences underwent transformations, e.g. number agreement that could be applied to single sentences(singularity transformations), then were joined into a single sentence by action of generalized transformation.

In later versions, these were handled by allowing the base to generate S nodes as components of other nodes(Fig.A-6). These nodes are expanded using the same rules as for the S node at the top of the tree. The result is a single deep structure tree for the entire sentence. A P-marker containing embedded S nodes is called a generalized phrase marker. Subsequently, the symbol $\bar{S}$ was introduced for a constituent containing a sentence along with a complementizer(for, that).

Abstract use of symbol. In most transformational grammars, symbols are assigned in whatever way makes the statement of the rules easiest. In Fig.A-7 (Passive transformation) a NP node was produced whose constituents are the word by + NP. This node bears little structural resemblance to the kind of NP generated in the base. However, this enabled the phrase "by . . ." to be manipulated by other rules as though it were an NP, making it easier to state certain transformations.

Lexical insertion. The transformational model described has two places where words might be inserted into the structure. If we think of the base component as having lexical insertion rules, e.g. "NOUN --> giraffe," the transformational component would begin with a tree having its lexical items included. These, though, would need to be more abstract than actual words, since it is only after the transformations that we can determine the exact surface form, including irregular verbs, contractions, etc. The other approach is to have a base that produces leaf nodes containing lexical categories and other features, but to let the transformations insert the words. Certain grammatical words, e.g. "by" in passive are inserted by transformations in almost all versions of the theory. For other words, in the standard theory it was assumed that there was a discrete phase of lexical insertion following the base generation and preceding all transformations. Only grammatical words could be introduced by transformations. The fact that the base structures are logical rather than syntactic provides a way of dealing with a number of other phenomena such as the analysis of adverbial modifiers.

Ordering of rules. Certain rules of grammar are postulated as being at least part of the knowledge every native speaker must have about his language. In grammars with generalized rewrite rules (unlike C-F grammars), it is possible for the order in which the rules are applied to have a significant effect on the outcome of a derivation. There may be a plausible derivation in which

$$\Gamma \circ \gamma \neq \gamma \circ \Gamma$$

i.e. rule γ is followed by Γ. But if rule Γ is applied first, it may change the structure so that rule γ is no longer applicable (see Appendix B-3). We explained the occurrence of sentence (2) above simply by ordering the process of subject copying before the rules deleting the subject NP, "you" (imperative).

As linguists examined the various transformations that were needed to describe English, they found tantalizing regularities in the consequences of choosing particular orderings. Much of the early excitement came from the feeling that grammarians were discovering a set of rules with a natural underlying order, which therefore must have some kind of psychological reality. Explaining the variety of new facts by using only those rules which are a necessary part of the grammar anyway brings us a little closer to what may be going on in the heads of native speakers concerning their knowledge of English. Figure A-11 illustrates a simple transformation ordering problem. The lines indicate specific ordering arguments. There is not an argument for every pair, but the combination of all pairwise orderings leads to a total ordering.

Dative Movement	(Optional)
Equi NP Deletion	(Obligatory)
Raising to Object	(Obligatory)
Raising to Subject	(Obligatory)
For Deletion	(Obligatory)
Passive	(Optional)
Agent Deletion	(Optional)
Reflexivization	(Obligatory)
Extraposition	(Optional)
It Deletion	(Obligatory)
Number Agreement	(Obligatory)
There Insertion	(Optional)
Tag Formation	(Optional)
Negative placement	(Obligatory)
Contraction	(Optional)
Subject-Auxiliary Inversion	(Obligatory)
Wh- Fronting	(Obligatory)
Affix-Hopping	(Obligatory)
Do Support	(Obligatory)

Figure A-11. Ordering of rules.

It soon became apparent that a simple linear ordering could lead to ordering paradoxes. This kind of problem arises only when there is an embedded S node, and transformations are applied both to it and the matrix sentence. This led to the postulation of "transformational cycle," in which rules are applied to the different S nodes in turn. There is still a linear ordering of the rules, but the entire sequence of rules is applied to each embedded S before any are applied to the S containing it. The grammar cycles through all of the rules once for each S node.

Computational Implications

There has always been an uneasy relationship between linguists working with the computational paradigm and researchers in Artificial Intelligence(AI), who adopt a more computational orientation.

One of the central thrusts of transformational grammar is the autonomy of syntax. The assumption is that the human mental capacities underlying syntactic competence are the result of specialized mechanism (the language faculty), that is not shared by other animals, and are to a large extent distinct from the rest of our mental processes. Thus the study of syntax is seen as relatively unique opportunity to isolate the operation of one component of mental processing and understand its special structure. Within AI, the goal has been the identification of the general principles and mechanisms that underlie all thought processes. Rather than look for specialized faculties, researchers formulate general theories of representation and problem solving that can be applied to a wide range of information processing activities. Any particular kind of competence is assumed to share the same basic structure unless there is explicit need to postulate otherwise. This difference in starting point has led to accusations from the two sides. Those working in TG argue that all of the proposed mechanisms in AI are so general that they cannot serve as a theory of the human mind at all; while AI researchers rejoin that transformationalists propose highly specialized "kludges" to account for properties of language that are the result of more general cognitive principles.

Along with the difference in assumption about generality and specificity, there is a deep difference in the way formalisms are used. In TG, the intuition grew out of mathematics and the physical sciences. A theory was a terse set of axioms that can be used to predict the data and not something to be programmed. One theory was better than the other because it is more restrictive--stated often as "less powerful." Much of the meta-theoretical discussion in TG deals with questions of how to limit the power of the formalism so that it can serve a more precise theory of the capacities it models. From this point of view AI models are hardly theories at all. Since the computational paradigm grew out of computer programming, the formal structures are seen as analogous to programming languages and data structures. They are good to the extent that they make it possible to model the observed phenomena of language use. Often a particular mechanism would be preferred because it is "more powerful" than its predecessors, since it allows it to cover more phenomena.

The most obvious difference between the two approaches is in the amount of attention paid to issues of the mental processes that go on when language is used. Distinction between competence and performance, as we saw, was one of the main theoretical bases of TG. Early in its development, the theory went in the direction of handling a wide range of syntactic problems with succinct formalism, at the expense of processing concerns. Generative grammar "generates" only in the most abstract sense and the formalism has little or nothing to say about what goes on when a person produces or comprehends an utterance.

The computational approach takes a different starting point. The information processing that goes on in the mind of a person using language is taken as fundamental. It is assumed that this process makes use of stored structures--such as grammar--that are not themselves procedural, but whose form is constrained by the way they are used in procedures of production and comprehension. This viewpoint seems to indicate that a characterization of abstract competence will inevitably fail to capture the appropriate generalities about language.

Computational Approach and Transformational Grammar

Although early attempts to find psychological correlates of the mechanisms of TG failed, it is still possible that some version of the theory may in the end serve as part of a comprehensive theory of language processing. There have been some attempts to apply formalism of transformational theory directly in the design of computer systems. This has been a difficult venture with debatable results although some researchers see it as a profitable path to follow.

If the overall transformational model does not serve as a psychological theory or as a basis for computer programming, it is quite likely that parts of the formalism will prove useful. Some computer systems make use of some kind of "transformational" procedure, often within a rather different overall framework; others have made use of formal devices developed in TG. A familiarity with the different versions of TG can serve as a valuable source of ideas for developing other theories and models.

The various amounts of variations that have been suggested for TG makes it of interest both for the variety of ideas and as an example of how a complex unified framework can serve as the basis for a broad field of formal explorations. Within linguistics, and generally cognitive science, TG is the formalism to which the greatest number of people have devoted a great deal of effort [CH57,65; AK75; BA74; WI83]. It is thus a well-developed formal mechanism. An understanding of its development including technical details provides a valuable perspective on what it means to develop a formal theory to account for a complex collection of data.

Indeed it has also become the language for characterizing problems. The fact that so much work has been done within TG makes it the lingua franca in which different phenomena have been described and discussed. A knowledge of the names given to many phenomena of importance is necessary for communicating within linguistics--even with linguists who work with different frameworks. Most of the attempts to compare the coverage of different computer grammars have used transformational terms in characterizing the data, even though the programs are not based on TG.

It has been typical for training in AI to exclude serious attention to the details of TG. There are reasons why, despite the differences, for any serious student of language within the computational paradigm to understand both the history and direction of current work in TG.

There are currently several research projects in which transformational and computational approaches are being brought together [WI83]. Work on realistic TGs is being combined with work on Augmented Transition Networks (ATN) which grew out of computational considerations. In the synthesis, some elements of each are preserved and the overall framework combines aspects of both. In order to understand the development and promise of this, a familiarity with transformational background and computational sources is necessary.

APPENDIX B

PRONOUNS and MORE TRANSFORMATIONS

B-1 Pronouns

Nonreflexive pronouns can occur almost anywhere

$$\left\{\begin{array}{l}\text{I}\\ \text{You}\\ \text{He}\\ \text{She}\\ \text{It}\\ \text{We}\\ \text{They}\end{array}\right\}\quad\text{will go}$$

or they may appear in indirect object position in slightly modified (or inflected) form as:

$$\text{He will write a note to}\quad\left\{\begin{array}{l}\text{you}\\ \text{him}\\ \text{her}\\ \text{them}\end{array}\right\}$$

(Note: If "him" and "he" refer to different people, the sentence is acceptable)

or in direct object position as

$$\text{I wash}\left\{\begin{array}{l}\text{you}\\ \text{him}\\ \text{her}\\ \text{it}\\ \text{them}\end{array}\right\}$$

The distribution of the reflexive pronouns can be described by the following set of rules:

(i) No reflexives may appear in the subject postion.

$$\text{(ii) - (ix)}\quad\left\{\begin{array}{c}\text{myself}\\ \text{himself}\\ \text{herself}\\ \text{itself}\\ \text{yourself}\\ \text{themselves}\end{array}\right\}\quad R\quad\left\{\begin{array}{c}\text{I}\\ \text{he}\\ \text{she}\\ \text{it}\\ \text{you}\\ \text{they}\end{array}\right\}$$

$$\left\{\begin{array}{c} \text{yourselves} \\ \text{ourselves} \end{array}\right\} \qquad \left\{\begin{array}{c} \text{you} \\ \text{we} \end{array}\right\}$$

where

$R = \{(x,y) \varepsilon PxP \mid x \varepsilon$ (direct, or indirect object), $y \varepsilon$ (subject)$\}$

$P = \{$pronouns: reflexive, normal$\}$

Thus

R = may appear as direct or indirect object iff the subject is list of pronouns appearing in (ii-ix).

The distribution of regular pronouns follows:

(x)

$$\left\{\begin{array}{l} \text{you} \\ \text{him} \\ \text{her} \\ \text{it} \\ \text{us} \\ \text{them} \end{array}\right\} \quad R \quad \{ \text{I} \}$$

(xi)

$$\left\{\begin{array}{l} \text{me} \\ \text{him} \\ \text{her} \\ \text{it} \\ \text{us} \\ \text{them} \end{array}\right\} \quad R \quad \{ \text{you} \}$$

(xii)

$$\left\{\begin{array}{l} \text{me} \\ \text{you} \\ \text{her} \\ \text{it} \\ \text{us} \\ \text{them} \end{array}\right\} \quad R \quad \{ \text{he} \}$$

(xiii}

$$\left\{\begin{array}{l} \text{me} \\ \text{you} \\ \text{him} \\ \text{her} \\ \text{us} \\ \text{them} \end{array}\right\} \quad R \quad \{ \text{it} \}$$

(xiv)

$$\left\{\begin{array}{l} \text{me} \\ \text{you} \\ \text{him} \\ \text{her} \\ \text{it} \\ \text{them} \end{array}\right\} \quad R \quad \{ \text{we} \}$$

(xv)

$$\left\{\begin{array}{l} \text{me} \\ \text{you} \\ \text{him} \\ \text{her} \\ \text{it} \end{array}\right\} \quad R \quad \{ \text{they} \}$$

i.e. Any nonreflexive pronoun may appear in direct or indirect object position so long as it is not identical with the subject of the sentence. All nonreflexives may appear in the subject position.

> Lemma: Anytime a sentence is generated in which there are two identical noun phrases, there is a rule which causes the second noun phrase in that sentence to assume its reflexive form.

Thus, in the deep structure we will generate(allow to occur) all the sentences of the form:

#I wash me

#We wash us(we)

#I will write a note to me

#You will write a note to you

(Note: # preceding a sentence or phrase means it is not acceptable)

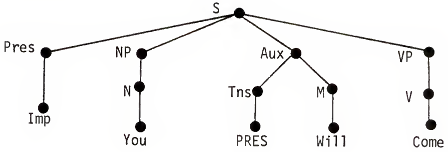
But because in each of these sentences there are two identical noun phrases, the rule(Reflexive), will automatically cause the second identical noun phrase to assume its reflexive form.

B-2 Imperative

Consider the sentence "Come!"

SD:	#	Imp	you	pres	will	X	
		1	2	3	4	5	
							--->Oblig
SC:		1	0	3	0	5	

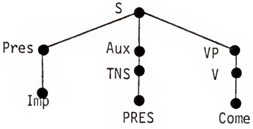
before:



Deep structure of come!

Imperative must apply since "you" is subject NP of the sentence.

After imperative:



both the subject "you" and the modal "will" have been deleted.

B-3 Reflexive and Imperative

Consider:

(Reflexive)

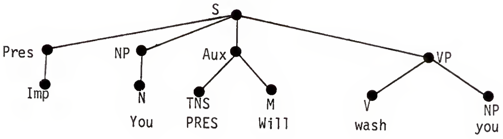
SD:	X	NP	Y	NP	Z
	1	2	3	4	5
					--->Oblig
SC:	1	2	3	[4] [+refl]	5

(Imperative)

SD:	#	Imp	you	pres	will	X
		1	2	3	4	5

SC: 1 0 3 0 5 --->Oblig

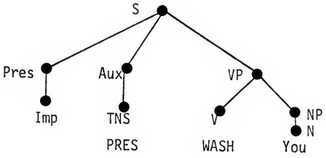
To order two(or more) transformation, consider: "Wash yourself" which has the following deep structure:



SD: 1 2 3 4 5

SC: 1 0 3 0 5 --->Oblig

Both reflexive and imperative can apply to this structure. Suppose imperative is applied first, then the structure is transformed to the following:



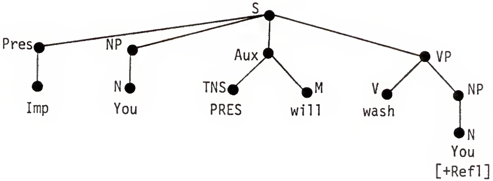
We note that the SD for reflexive cannot be met since only one NP is left. This structure will eventually give an unacceptable "*Wash you!"

Consider the same structure again and apply reflexive first

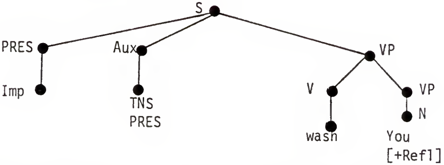
SD: 1 2 3 4 5

```
                                        --->Oblig
SC:   1    2    3  [ 4  ] 5
                   [+refl]
```

after Reflexive:



The imperative can still be met after reflexive is applied. The structure then becomes:



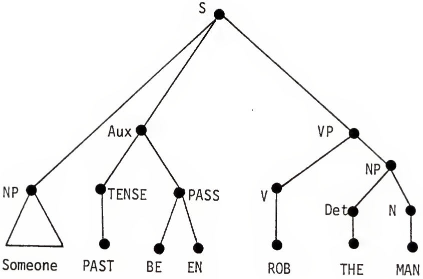
Another transformation may apply--Affix Hopping.

B-4 Agent Deletion

Let us relax the requirements of the Passive rule, allowing optional preposition(when applicable) to appear in its SD

(NP - Aux - V - (P) - NP)

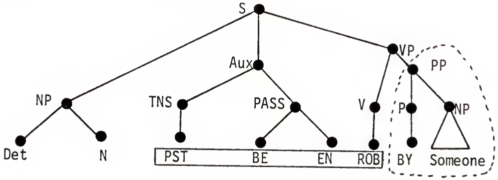
The sentence "The man was robbed" has the following deep structure:



SD: 1 2 3 4 5

SC: 1 4 3 0 5 by 2

Only passive applies to create an agent which may be deleted by an Agent Deletion. We cannot delete what has not yet been created. Since the tree has been analyzed to meet the SD for Passive, the structure becomes:

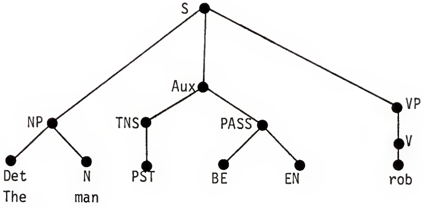


Agent Deletion:

SD: X passive V - by NP - Y
1 2 3
--->
SC: 1 0 3
where 2 = by + some { one }
{ thing }

Now the Passive has created an agent which may be deleted,

Agent Deletion applies optionally:



After Affix Hopping we have "The man was robbed."

APPENDIX C

C-1 ATN PROCEDURES AND A GRAMMAR

This section describes some of the procedures (non-deterministic) in using an ATN grammar. We have followed the DL notation of Winograd.

```
+------------------------------------------------------------+
|                AUGMENTED TRANSITION NETWORK                |
+------------------------------------------------------------+
 Describes: a template for the constituent structure of a
            composite category
 Includes: label, state, arcs
  +--------------------------------------------------------+
  |                          ARC                           |
  +--------------------------------------------------------+
                       A set of arcs
   Describes: a transition between states
   Includes: label, start and end states, initialization,
             condition, action
     Label: depends on the arc
     Initializations: a set of initializations, may be empty
     Conditions: a set of, may be empty
     Actions: a sequence of, may be empty
  +--------------------------------------------------------+
  |                         STATE                          |
  +--------------------------------------------------------+
                   A set of states
   Describes: a state in the network
   Includes: label, incoming and outgoing arcs
     Label: unique to a state, in a grammar
     Incoming arcs: arcs with this state as the ending
                    state
     Outgoing arcs: arcs with this state as the starting
                    state
  +--------------------------------------------------------+
  |                       ROLE NAME                        |
  +--------------------------------------------------------+
        Distinct from other role names in the network
   Describes: name for a role register
+------------------------------------------------------------+
```

Figure C-1. Augmented transition network.

```
+-----------------------------------------------------------+
!               Parsing with an ATN grammar                 !
+-----------------------------------------------------------+
 Required: to produce a structure representative of a
            parse of an input sequence in a language
            defined by the grammar
 Method: - create a register table for the net labeled S
           with no initializations and a null configuration
         - match the net against the input, starting in the
           initial state, with the register table
         - if the match succeeds and the position returned
           is one greater than the length of the sequence,
           return the resultant node from the match.
           Else fail.
 Result: a register-structure node (Chapter 3)
+-----------------------------------------------------------+
```

Figure C-2. Top-down parsing with an ATN grammar

```
+-----------------------------------------------------------+
!                 Create a register table                   !
+-----------------------------------------------------------+
 Required: to create an initialized table of registers
 Inputs: an ATN network; set of initializations
 Method: - do, for each feature dimension of the net:
             add an entry whose key if the name of the
             dimension and whose entry is the default
             for the dimension.
         - do, for each of the initializations:
              - find value in the configuration (which
                describes the situation at a particular moment
                in a parse; it includes the register table
                for the constituent being built, the current
                network, state, position, and node most
                recently parsed.)
              - add result to register table under the key.
              - Return register table.
 Result: a register table
+-----------------------------------------------------------+
```

Figure C-3. Creation of register tables.

```
+-----------------------------------------------------------+
!                  Procedure to match an ATN                 !
+-----------------------------------------------------------+
 Required: to test for a match
 Inputs: ATN; input sequence; state; position in the
          sequence; register table
 Configuration: a configuration; initially new with:
                the state, the position, most recent node
                empty, and registers
 Method: Repeat,
       - choose an outgoing arc of the state of the
         configuration
       - match the arc against the input, using the
         configuration
       - if the match succeeds, assign the result as the
         configuration; else fail
       - if chosen arc was 'send' do:
         - create a new node with:
           - label as the label of the network
           - registers as the registers of the configuration
         -return this node as position of configuration

  Procedure:
           match an arc against the input
   Input: an arc; a sequence, and a configuration
   Result: a configuration

 Result: a node; an integer representing a position in
         the input sequence
+-----------------------------------------------------------+
```

Figure C-4. Procedure match an ATN

```
+------------------------------------------------------------+
|                        Category Arc                        |
+------------------------------------------------------------+
 Describes: a transition by matching an input
 Includes: a label which is the name of a lexical category;
           starting and ending states; conditions and
           actions
 Needs: and ATN and a dictionary
 Procedure:
           match a category arc against the input
  Goal: produce a new configuration
  Inputs: a category arc; input sequence; configuration
  Method:
      - if position from configuration is greater than the
         length of sequence, fail
      - choose a word sense in the dictionary entry for the
         word at the position and assign it as the word
         sense
      - if category of the word sense is not the category,
         fail.
      - create a new node with:
        - label as the category
        - registers as an empty table
      - for each feature dimension of the category, fill
         the registers of new node with:
        - key as the name of the dimension
        -entry as
          - if there exist an entry for the dimension in
             the word sense, then that entry
          - else, the default for that dimension
      - create a new configuration, as a copy of old with:
        - most recent node as the new
        - state as the ending state of the arc
        - position as (position of old configuration)  + 1
      - test conditions of arc of new configuration; if
         they fail, then fail
      - execute the actions of the arc on the new config.
      - return new config.

  Result: a configuration
+------------------------------------------------------------+
```

Figure C-4a. Procedure match a category arc.

```
+-----------------------------------------------------------+
!                         Seek Arc                          !
+-----------------------------------------------------------+
  Describes: a transition by a recursive match in a network
  Requires: an ATN grammar; network where the arc appears
  Includes: label which is the label of a network of the
             grammar or a state in one of the networks;
             starting and ending states of the network;
             initializations, conditions, and actions
  Procedure:
            match a seek arc against a sequence
    Goal: produce a new configuration
    Input: a sequence, seek arc, and a configuration
    Working Structures:
       - a network, if the label of the arc is:
           - label of a network, then that network
           - label of a state, then the network comprising
              that state
       - a state, if the label of the arc is:
           - label of a state, then that state
           - label of a network, then its initial state
       - an integer which is the position from the config
       - a register table resulting from the creation of
          registers for the network with the initializations
          of the arc using the configuration
    Method:
          - match the network against the input beginning at
           the position, starting in the state, with the
           registers
          - create new configuration as a copy of old with:
            - most recent node as the node returned by the
               match
            - state as the ending state of the arc
            - position as the position returned by the match
          - test the conditions of the arc on the new config.
             If they fail, the match fails.
          - invoke the actions of the arc on the new config
          - return the new configuration

    Result: a configuration
+-----------------------------------------------------------+
```

Figure C-4b. Procedure match a seek arc.

```
+------------------------------------------------------------+
|                          Jump Arc                          |
+------------------------------------------------------------+
  Describes: a transition that uses no input
  Needs: an ATN
  Includes: label 'jump;' starting and ending states;
            conditions and actions
  Procedure:
           match a jump arc
     Goal: produce a new configuration
     Input: jump arc; a configuration
     Method:
        - create new configuration as copy of the old with:
            - state as the ending state of the arc
        - test conditions of arc on the new configuration.
           If they fail, the match fails.
        - invoke actions of the arc on the new configuration
        - return the new configuration

     Result: a configuration
+------------------------------------------------------------+
```

Figure C-4c. Procedure match a jump arc.

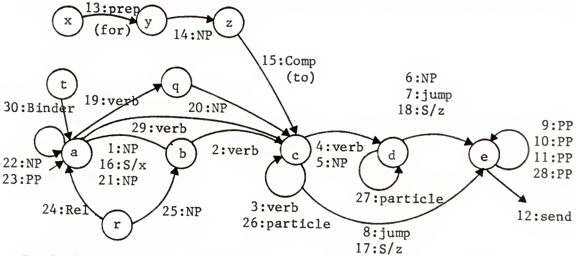
```
+------------------------------------------------------------+
|                          Send Arc                          |
+------------------------------------------------------------+
  Describes: a transition that terminates a network
  Needs: an ATN
  Includes: label 'send;' starting state; conditions and
            actions
  Procedure:
           match a send arc against a sequence
      Goal: produce a configuration
      Input: a sequence, a send arc, a configuration
      Method:
           - test conditions of the arc on the
              configuration. If they fail, the match fails.
           - invoke the actions of the arc on the
              configuration.
           - return the configuration

      Result: a configuration
+------------------------------------------------------------+
```

Figure C-4d. Procedure match a send arc.

```
+------------------------------------------------------------+
|                         DICTIONARY                         |
+------------------------------------------------------------+
  Describes: a dictionary of lexical categories and
             features of individual words
  Includes: a set of lexical categories, and word entries
  Classes:
  +--------------------------------------------------------+
  |                   Lexical Category                     |
  +--------------------------------------------------------+
    Describes: a category for word class
    Includes: name, feature dimensions
       A word belongs to a lexical category if there
       exists a word entry in the dictionary with the
       word as its word, consisting of a word sense
       with the category as its category.
  +--------------------------------------------------------+
  |                      Word entry                        |
  +--------------------------------------------------------+
    Describes: an entry for a word
    Includes: a set of word senses, a word
       A word can have senses whose categories are
       different, can have senses with the same
       category and different features.
  +--------------------------------------------------------+
  |                      Word sense                        |
  +--------------------------------------------------------+
    Describes: a particular sense for a word, which may
               specify choices for each feature dimension
               of the category
    Includes: a lexical category;feature table whose keys
              are the names of feature dimensions of the
              category, with each entry a choice for the
              dimension
+------------------------------------------------------------+
```

Figure C-5. Dictionary for ATN grammar.



Includes: Subj, q-e, d-o, i-o, m-v, binder, mods, aux
Feature dimensions: Mood: Decl, Int, Imp, whRel, Bound Rel

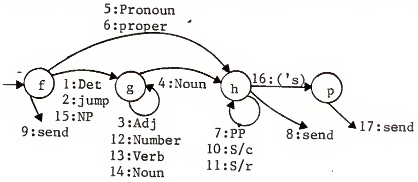
```
Init, conds, actions:
Arcs 1-25 as in Chapter 3.

S_c26c
      A: m-v <- dictionary entry
      C: m-v + # e dictionary
S_d27d
      A: m-v <- dictionary entry
      C: m-v + # e dictionary
S_e28e
      A: m-v <- dictionary; d-o <- (#)Prepobj
      C: d-o ≠ 0; m-v + (#)Prep e dictionary
S_a29c
      A: Mood <- Imp; Subj <- dummy NP with head=you;
         m-v <- #
      C: Mood = Decl; (#)form = Inf
S_t30a
      A: Mood <- Bound; Binder <- #
```

Figure C-6a. The Sentence network.

```
+------------------------------------------------------------+
|          Conditions for subject-verb agreement              |
+------------------------------------------------------------+
       Either: (m-v)type = modal or (m-v)form = Past
       or (m-v)form = 3rd-Sing and (Subj)Num = Sing and
        (Subj)Pers = 3rd
       or (m-v)form = Inf and either (Subj)Num = Plu
        or (Subj)Pers ≠  3rd
+------------------------------------------------------------+
|               Conditions for verb sequence                  |
+------------------------------------------------------------+
       If (m-v)type = Modal or Do => (#)form = Inf
       If (m-v)type = Be => (#)form = Pres-Part
       If (m-v)type = Have => (#)form = Past-Part
       If (m-v)type = non-aux => fail
+------------------------------------------------------------+
|               Conditions for transitivity                   |
+------------------------------------------------------------+
       If i-o ≠  0 => (m-v)Trans = Bitrans
       If i-o = 0 and d-o ≠  0 => (m-v)Trans = Trans
       If d-o = 0 and i-o = 0  => (m-v)Trans = Intrans
+------------------------------------------------------------+
```

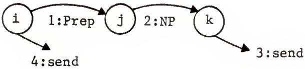
Figure C-6b. The Sentence network.



Includes: head, describers, qualifiers, determiner
Feature dimensions: Number: Sing, Pl; default - empty.
Ques: Yes, No; default - No. Case: Subj, Obj,
Poss; default - empty. Person: 1st, 2nd, 3rd;
default - 3rd

```
Init, conds, actions:
Arcs 1-9: as in figure 4-10
NP_h8
        A: Case <- (Head)Case.
NP_h10h
        A: Qual <= #
        I: Mood <- Rel; m-v <- dummy (be); Subj <- !copy
NP_h11h
        A: Qual <= #
        I: Mood <= whRel; hold <= !copy
NP_g12g
        A: Desc <= #
NP_g13g
        A: Desc <= #
        C: (#)form = Pres-part or Past-part
NP_g14g
        A: Desc <= #
        C: (#)Num = Sing
NP_f15g
        A: Det <- #
        C: (#)Case = Poss
NP_h16p
        C: head ≠ pro
NP_p17
        A: Case <- Poss
```

Figure C-7. The NP network.



```
  Includes: Prepobj, Prep

  Initializations, conditions, actions:

   PP_i1j
           A: Prep <- #
   PP_j2k
           A: set Prepobj to #
           C: (#)Case = Obj
   PP_k3
           no init., cond., or actions
   PP_i4
           A: empty and return hold
           C: hold is PP
```

Figure C-8. The PP network

C-2 Changes to the formal mechanism

The register structure described employs a register table associated with each node. A special hold register was also introduced to alleviate certain inadequacies in the structure. The hold register does not form part of the final node structure produced by the parse, and needs to be kept separately. A "hold table" can be assigned to the configuration.

An item in hold can be used by an embedded phrase and it is necessary for the hold register to be passed when a seek arc is taken. This should be reflected in Figure C-4b in its call to "match a network . . . ;" which is its current hold table.

A held item is usable only once. The contents of a hold register are destroyed when it is used to set some other register. Thus the entry in the hold table is removed. To communicate this fact the entire table is passed down. If an embedded phrase removes the item form the table, the higher(passing) node will have an empty table after the lower network returns.

With other registers, any action or initialization, causes the entry in the register table to be changed, but the table, with all of its other entries remain the same. On action or initialization of the form "Set HOLD to ...," the entire hold table is replaced by a new table with the item as its entry. As a result of this kind of displacement, there can be different hold registers associated with different levels of the structure being parsed(as in nesting), achieving the effect of a hold list.

In the networks presented, the initializations ensure that the relative clause will have its own hold register, (not an inherited one), the emptying ensures that an item will be used only once, and the conditions that the hold register be empty(INTERROGATIVE and Wh-RELATIVES's) on the S_e12 arc guarantees that a held item will be used up within the constituent in which it was put onto hold.

C-3 Relative clauses

Here we consider sentences which may have one or more sentences embedded in them. Embedded relative clauses typically has a wh-word followed by the rest of the relative clause.

If there were no grammatical transformations, we might ask, what kind of phrase structure rules would be needed to generate the kind of structures?

A suggestion might be:

NP --> (Det) + (Adj) N (Modifier)

Modifier --> Wh-word S

The relative clause formation rule is stated as:

Relative clause formation -

```
SD:   X - [ NP  [ Y - (P) - NP - W] ] - Z
      1     2     3    4    5    6      7

SC:   1     2     4  [ 5  ] 3    6      7
                     [+pro]
                     [+wh ]
```

condition: 2 = 5

The NP structure for Relative clause formation is

```
                NP
                /\
               /  \
       [   NP     S    ]
       [         /\    ]
       [       [S--S]]
       [               ]
      NP              NP
```

Copy node in relative clauses

Figure C-9 illustrates the use of COPY in a structure, to deal with the problem of relative clauses(arc NP_h10(s/c)h). This is indicated by the phrase "a copy based on ↑ " for the value to be put in the register.

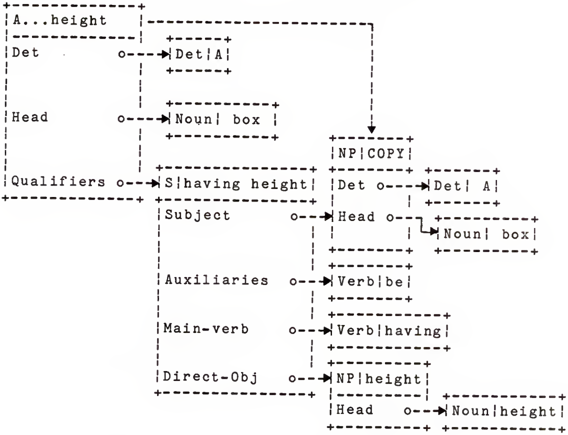


Figure C-9. A structure with copy.

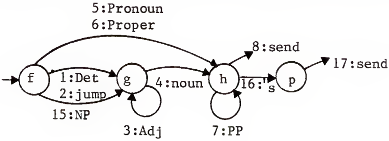
C-4 Additional syntactic issues

The basic mechanism of ATN described herein are not sufficient for handling all of the phenomena of natural language. As with other syntactic formalism, there is a "basic theory" and an ever-increasing collection of changes and extension to cover more of the data. There are a number of issue raised by English which can be handled with existing mechanism by additions of new arc and states to the grammars of chapter 3. Others call for more radical modifications of the formalism.

For practical issues of ATN grammars, one picks out a subset of structures that will be handled, ignores the others and includes ad hoc methods to deal with those structures which has not been addressed linguistically.

C-4.1 Case agreement

We have introduced actions to check for person and number agreement. But there are other kinds of agreement in the language. English has a limited form of case marking that applies to pronouns but not other forms of NP. This can be handled without new arcs by addition of a feature dimension "CASE" with choices ACCUSATIVE(OBJECTIVE), NOMINATIVE(SUBJECTIVE), and POSSESSIVE. An NP is initialized with no case choice. The choice is set by the send arcs(the normal exit will set it to the case of the head - which may be noun, proper noun or pronoun - and the possessive send arc sets it to POSSESSIVE. The pronouns her, him,them,us, etc., are marked OBJECTIVE, she, he, who, I, etc., are SUBJECTIVE, and my, your, her, etc., are POSSESSIVE. Each common noun and proper noun has senses for both SUBJECTIVE and OBJECTIVE, since this distinction is not reflected in the form of English noun. Some pronouns(e.g. it, her) have senses with different features. The changes in the NP network are shown in Figure C-10. Arcs that parse an NP must



Includes: Det, Descr, Quals, Head
Feature dimensions: Person: 1st, 2nd, 3rd; default-3rd
Number: Sing, Plu; default- empty
Case: Obj, Subj, Poss; default- empty
Question: Yes, No; default- No

Init, Conds, Acctions:
Arcs 1-9: as in Figure 3-10; 15-17: as in Figure C-7

Figure C-10. Possessives in NP net.

each have a condition specifying the appropriate feature. Arc S_a1b, which parses the subject of an ordinary sentence, has a condition "the CASE of # is SUBJECTIVE," while S_c5d and PP_j2k have "CASE of # is OBJECTIVE." Arcs responsible for dealing with simple relative pronouns and questions allow either case, because agreement will be checked by the arc using the NP that removes the element from the hold list.

C-4.2 Verb sequence

The network will accept a main verb prefixed by a sequence of auxiliaries - which may include nonsensical clauses. The restriction on possible ordering are basically the conditions on S_c3c arc. However, there are a number of complexities on what order things occur in.

There is a need to account for multi-word verbs made up of a verb and a particle(e.g. call up) or a verb in combination with a preposition(e.g. call for). The grammar as is will allow for the latter but the transitivity features will not be correctly handled, especially with non simple intransitive verbs. Verb-particle combinations are more complex for a particle is allowed before or after an object(called up mother/called mother up).

Basically, the solution is to allow extra arcs to parse particles and PPs and allow for actions on both arcs that modify the m-v register to reflect the combination. They will need to check that the verb and particle or preposition can be combined(e.g. up is not a particle in ". . . up the stairs," as evidenced in the ungrammmaticality of "*. . . the stairs up.") The dictionary needs to be extended to include entries for such pairs, to have the appropriate transitivity features and a definition reflecting the meaning of the combination. The send arc of S network would then have to check that the special senses appeared only with the appropriate particle or preposition. Another solution would be to have these unique entries unavailable when a normal category arc is taken with category VERB, but to have them available to replace the verb entry when the particle or preposition is found. The unique entry along with its own feature choices is put in place of the previous m-v, and would also provide a basis for procedures dealing with the meaning of the sentence. The formalism for conditions and actions will also need to be extended beyond that of Chapter 3 to allow the appropriate entry to be found and used--Figure C-11.

Arcs 26-28 of Figure C-6

Figure C-11. Addition to network to handle multi-word verbs.

The word sense resulting from traversing one of these arcs will not be combinable with another particle or preposition. Consequently, one combination will be allowed in parsing an S.V. However, in order to forbid a pronoun following a verb-particle pair, we force additional condition in S-c5d arc: if M-V is a verb-particle pair, then the HEAD of # is not a pronoun.

C-4.3 Modifiers

The S network as described not only does not accept modifiers at the beginning, only at the end, but also does not include any of the other many types that can appear in both places (e.g. -ing clause, Bound clause, Time NP, (for-)to- clause, adverb, adverb group). We can create loops in states S.a and S.e for each of these and include action to append the result to the MODIFIER register. This would, however, pose some problems.

Subject Initialization. The loop for a modifier -ing clause would have label S.c while that for a (For-)To-clause would have label S.z. Arcs already exist for which each of this labels--NP_h10(s/c)h and S_c17(s/z)e. In those cases there was a register that could be initialized to the subject of the embedded S, while in the case of the sentence modifier there is none. Often the subject is the same as the (surface) subject of the S, but can also be the object. If the modifier appears at the beginning, we have to initialize it to something that has not been parsed yet. If it appears at the end, we need to account for all the possibilities. This is impossible without significantly revising the ATN formalism.

The question then is how much detail should be ascribed to syntactic formalism and how much should be be left to the semantics. We recall that one of the basic assumptions underlying the networks and of most work in TG, is that the syntactic component should account for intuitions. It is possible to assume otherwise and allow the semantic component instead, making it easier to deal with above cases, where the syntactic connection is unclear or missing. This can be done properly if the syntactic and semantic components interact closely, so that the kinds of connections we have demonstrated earlier are properly accounted for.

Relative Clauses and Questions. NPs and adverbs can be used as question elements and can have relative clauses. The loop for parsing on initial adverbs would need to note if it is a question form (appropriately marked in the dictionary) and if it is, set the hold register and question-element and mark the sentence as INTERROGATIVE. Either of the loops for a time NP will automatically pick up an item from the hold list to handle the other cases.

Overgeneration. The network with the added loops accepts many sentences that seem too contorted to be really acceptable. It could be overly restrictive to forbid certain modifiers of a kind at certain positions in a clause when they seem perfectly acceptable in other sentences. In modifying the network to accept a wide range of structure, ATN grammars are usually written in a style that accepts many questionable constructs in order to have the generality to handle sentences intended. The problem is overgeneration--unless the networks are made very complex, they will accept sequences that are not grammatical as well.

Adverbs can also appear in the middle of a sentence. Adverb loops which append to the MODIFIER register will have to be added to states S.b and S.c. An adverb however, cannot be the last thing parsed in state S.c. There are ways this can be handled and they serve to illustrate the choices faced in developing an ATN grammar.

In general, we avoid complicating the basic formalism with special devices if the same effects can be achieved with existing mechanism. This allows for simpler implementation.

Additional mechanism. This mechanism would be used to keep track of the previous thing parsed. The conditions on the arcs leaving S.c (i.e. pre '#' constituent) can include a check that it not be an adverb.

Additional feature registers. A register may be added with a feature dimension ENDS-WITH-ADVERB with choices YES/NO. This would be set with the arcs that parse verbs and adverbs and could be also checked in conditions. Some grammar writers freely create such registers that do no have any significance for the final register table resulting from the parse, but are useful in controlling it. Thus treating ATN more as a programming device instead of as a structural grammar.

Additional role registers. In a similar manner, we could add a role register LAST-VERB-OR-ADVERB that is set by the same arcs. The conditions then is to check the word class of its contents rather than a YES/NO feature.

Additional State and Arcs. A solution may be to create a more complicated network in which state S.c is replaced by multiple states as in Figure C-12.

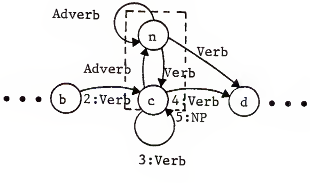


Figure C-12. Node expansion for adverbs

When an adverb is found, the parser goes to state S.n, remaining there as long as adverbs are found. On the next occurrence of a verb it escapes to S.c or S.d (if it is the main verb of a passive.) Once in S.c it can find any number of additional verbs before either encountering another adverb or proceeding on to the end of the sentence. This expansion allows verbs and adverbs to be mixed in an arbitrary order, provided the last thing is a verb.

this solution is typical of the kind of complication that is added to the grammar to account for syntactic subtleties. It is also typical of a case in which it is unclear what the grammar should allow. As it is, it will accept multiple adverbs between verbs, which may be alright, but may also allow questionable sequences. There are places in the grammar where it is impossible to pin down the data.

The adverb not only differs from others in that it cannot appear in a double negative; its contracted form "-n't" appears where other verbs cannot. A NEGATIVE feature dimension is added not only to S and dictionary entries for adverbs and verbs (contractions like can't are not treated as negative verbs), but also to NP and to entries for qualifiers (no) and pronouns (nobody). The register in each network needs to be set to NEGATIVE when such a constituent is parsed--prevents appearance with negative NP--and conditions need to be added to forbid this from happening more than once (prevents multiple uses.) This involves putting new actions and conditions on a number of arcs.

C-4.4 Structure of the NP

The existing NP network can deal with common NPs and those whose head is a proper noun or pronoun. It accepts PPs and two kinds of relative clauses following the head, even if the head is a proper noun or pronoun. It also has a special arc to deal with the hold list. However, it is incapable of handling the variety of modifiers preceding the head even in ordinary phrases. These may include past or present participle verbs, nouns, numbers, and words in special category such as "last."

Again, we may add additional arcs to handle this as in Figure C-13.

Loops are added for each kind of modifier, allowing for an arbitrary mixture following the determiner, but preceding the head. The action on each is "Append # to DESCRIBERS," and conditions to check for appropriate form (e.g. plural noun or infinitive verb not allowed). This seem general enough but allows unacceptable combinations. Ordinarily numbers are expected first, followed by adjectives and verbs, then nouns. However, this is not always the case. The rules for ordering modifiers in an NP are quite difficult to describe in any formalism.

It seems again natural to throw more states and arcs at the solution. But any ordering constraints that may be

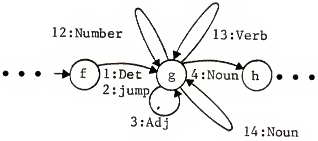


Figure C-13. Addition of arcs for NP modifiers.

specified by states and arcs can be done instead with conditions and actions, with the proviso that appropriate registers be added. There have been other proposals (e.g. Lackoff's cognitive grammar) to have each network consist of a single daisy, with all of the syntactic details embedded into the actions and conditions of the arcs. In general, the grammar offers two basic solutions to the problem of ordering. The structure may be a chain in which the sequence is implied by the order of states and arcs; or it may be a daisy structure as above in which loops can be taken in any order, but ordering constraints appear as conditions. Thus, if we would have a number first or following a single modifier that is not a noun, we may put a condition on the NP_g12g arc that would check the contents of the DESCRIBERS register to see that it is indeed empty or that there is no more than one element of the right kind.

One other consideration is that of a sequence of nouns. The NP_g14g arc parses a noun and appends it to the describers while the NP_g4h arc parses the head. A sequence of nouns will be successfully parsed if all but the last is consumed in the loop, and escapes with the last on the other arc. In building a deterministic parser, there are often mechanisms to avoid the creating of ultimately unprofitable paths. If the last is taken by the loop, progress is stopped, whereas if the loop is escaped prematurely the left-over nouns will block continuation.

Let us examine this kind of a loop structure further with a view to comparing it with the structure used in parsing a sequence of verbs. We have a sequence of words (either verbs or nouns), which must contain at least one entry. In the structure above, there is a loop followed by an arc (L->A), whereas in the S network, there is a single arc followed by a loop (A->L).

In deciding which structure to use, one must consider the associated actions and conditions. For instance, if the first item is to be saved in a separate register while subsequent items are combined the A->L structure allows the independent actions on the arcs. In the above (NP) case, all the nouns but the last are to be appended so L->A structure is used. With the verb sequence, it is the last element (m-v) that need be saved separately, yet the A->L scheme was chosen in the network. This is because of the need to check the subject-verb agreement and the proper sequence of auxiliaries and the verb forms. Now, the agreement check applies only to the first verb and does not apply at all in case of entry in S.x and S.z; also the auxiliary sequence check applies in parsing all but the first verb. These statements are more easily effected by having a separate arc for the first verb (A->L). The main verb will be set to the last verb by resetting it each time a verb is consumed, appending the previous value to the auxiliaries. Thus the final contents of the auxiliaries and m-v are just as though the L->A structure was used.

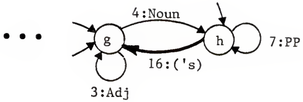
While we have accounted for the foregoing phenomena, the network still does not cover the full structure of the NP. It neither deals with predeterminers (e.g. "all . . ."), quantifiers("every day"), partitives("some of the time"), quantified pronouns("everything she says"), nor with details of the number agreement among determiner, numbers, quantifiers and nouns. It also does not deal with possessives.

C-4.5 Possessives

Let us consider the NP network of Figure C-10 in which loops exist for DESCRIBERS and CASE registers from the previous discussions. A possessive NP e.g. "my," "the professor's" is allowed on the NP_f15g arc in place of the determiner. A possessive pronoun if marked in the dictionary as such will be handled by the action on the pronoun arc NP_f5h; and the NP_h16('s)p arc will create a possessive NP out of an ordinary one by matching 's. We need to add a separeate NP_p17 send arc because if the arc that matches -'s is returned to the NP.h additional modifiers could be tagged on. This send arc sets the CASE to POSSESSIVE, instead of to the case of the head. Thus, this grammar will allow modifiers to be parsed before the -'s. The conditions enumarated earlier for dealing with case on arcs that use an NP will prevent a possessive NP from being used in any other place than on the NP_f15g arc. To handle possessives of the like "mine," "yours," further additions would be necessary.

This procedure seems satisfactory enough, but we observed that the first state in the network has an outgoing NP label. This poses a good candidate for left recursion. There are problems for some TLR parsers in handling left-recursive constructions; and we noted earlier that the ATN formalism is oriented toward TLR parsing. A left-recursive grammar can cause a top-down parser to go into an infinite loop. The setting of registers and use of their contents in specifying conditions must take place in the proper order, or the results will be totally garbled. Woods has shown that it is possible to eliminate direct left and right recursion and optimize the network using finite state optimization techniques. From the foregoing, either special mechanisms must be added to the parser, or other means must be sought to handle possessives.

Let us consider a new arc that operates after a head has been found, parses a -'s, and returns to the state normally reached after a determiner has been found, as a specialized way of handling possessives. Here we have eliminated the NP and send arcs. It has an associated action that creates a node for the NP as parsed so far and puts it



into the determiner register. In addition, it sets all of the other registers--both feature and role--back to the values they would have at the beginning of an NP. This operation is a far cry from the mechanism we described in the formal definition. We need to create a new node, then create a new configuration for the network that called it.

This is a more complex solution than the previous from a linguistic point of view. However, the practicality and feasibility it offers for TLR parsing is justified. This kind of specialized manipulation of the grammar in light of processing considerations pervades the building of practical systems.

C-4.6 Other clause structures

Imperative. A clause can function either as an independent constituent of a discourse or as a constituent within other clause or group. There are other forms of sentences (embedded and top-level) that are not covered by this grammar. Some may be achieved by the addition of a few simple arcs. Imperative like "Draw the circle" is possible with the addition of a simple arc S_a29c. Instead of the usual subject-verb pair, the sentence begins with a verb of the infinitive form (as in Be sharp) and the subject is an implicit "you." Again, we use a dummy node inserted by an action, where the corresponding imperative transformation would delete an item of that form.

```
C: The form of # is infinitive and mood is declarative
A: Set subject to a dummy NP with Head 'you'
   Set m-v to #. Set mood to imperative
```

The check on the mood will prevent this arc from being taken within a question, relative clause, or other form of embedded S. The explicit insertion of "you" as the subject (see Appendix B-2), provides a structure containing the appropriate information for an analysis of meaning and also make it possible to handle syntactic phenomena like tag questions, wherein some other constituents have to agree with the subject:

He promised to come, didn't he? .....(i)

Bound clauses (that you did it), can be handled by addition of state S.t in which they would begin

```
S_t30a
      A: Set Binder to #. Set Mood to Bound
```

A new role register is added to hold the BINDER (while, that, etc.) and a mood choice BOUND to indicate the kind of S and prevent a binder from being followed by an imperative verb, question form etc. One arc leading out of S.a that has a mood condition compatible with BOUND is the standard subject arc S_a1b with associated condition "MOOD is not INTERROGATIVE."

The addition of arcs for bound clauses is complex. Among other things, they depend on the particular binder.

For example the subject of a sentence can be a bound clause beginning with "that" or "whether" but not "while." There are other forms of embedded S and some can be parsed with the network. A form like " . . .that he be there," has a verb in an unusual form and requires modifications to the net. It is relatively easy to modify the network to accept a wide range of structures but again the problem is that of overgeneration--unless the networks are made complex, they will accept ungrammatical sequences.

In the presentation of questions and relative clauses in Chapter 3 we accounted for clauses in which the first element is a pronoun. There are NP or PP clauses that contain "which," "whom," or "whose" embedded within it. There is a phenomena known as "pied pipping" ((ii) below), in which material from the rest of the clause can be embedded within the initial phrase:

A girl the likes of which you have never seen ....(ii)

These can be handled with an extension and use of hold registers, in which the elements can be used by an arc that parses a relative pronoun when it occurs in the initial phrase of a wh- relative clause. The held items can be picked up to match a relative pronoun, and when the element is complete, it goes to the hold list to be used to fill a gap later in the clause. The gap may be a PP, rather than an NP. This will require more than a single hold-register (-list), since the two different ways of using held items--to match a relative pronoun or fill a gap--must be kept separate.

C-4.7 Transitivity

Transitivity is a much more complex phenomenon than as presented. Only the simplest kinds have been handled directly by the formalism. We have assumed that each verb may be classified as one or more of INTRANSITIVE (followed by no objects), TRANSITIVE (one object), BITRANSITIVE (two objects). The grammar needs to be modified to account for complements that appear after the verb but are no ordinary objects:

They drank the bottle DRY .....(iii)

It must handle verbs like "put" which demands a locational object in addition to a direct one:

*The student put the book .....(iv)

and others that take apparently ordinary objects but do not allow passive voice. Those systems that deal more with it in a comprehensive way combine ATN with some kind of case grammar analysis--Chapter 4.

C-4.8 Comparatives, conjunction, and ellipsis

There are other classes of phenomena that present difficulties for syntactic formalism, including ATN grammars. Among these are conjunction:

I drew a circle and painted it red .....(v)

comparatives:

The circle is larger than the box is tall .....(vi)

ellipsis (the leaving out of implied constituents) as in the response:

"John thinks C" to "What language does she prefer" .....(vii)

It seems as though conjunction might be handled by adding more arcs to the grammar, but there is a drawback since they can occur almost anywhere in the structure. Intuitively, we may decide to start parsing a new phrase, when a conjunction is encountered, that matches some phrase that we are in the process of parsing, and ultimately combine the two as a conjoined constituent; and if there are gaps in the new one, fill them in with the corresponding items from the old. Several systems have had ways of implementing such instructions, allowing the presence of conjunctions in the input to trigger special operations rather than being parsed by the ordinary operations of arc matching. However, they have not been able to achieve broad generality. Difficulties lie in the manner of specifying appropriately how the system should look through the set of phrases it has been parsing and decide what is appropriate to match. Often the section of the input that is matched is a partial constituent or one with gaps.

From a formal syntactic point of view, we need to put in place a mechanism that assign appropriate structures to sentences and reject nonsentences. From a practical point of view, conjunction raises another issue. It often leads to large amounts of potential ambiguity, since there are a number of different ways in which the structures following conjunction could be matched against the ones preceding it.

If a grammar is built to accept all of the possibilities, every appearance of conjunction will create an explosion of paths for the parser to follow, and the number of possibilities grows exponentially with more than one conjunction appearing. Hence, in practical systems, there is a tendency to limit the generality on the conjunctive mechanism in favor of using criteria based on meaning to limit the exploration to a small number of paths.

Comparatives present similar problems in that they are also combined with ellipsis as with:

I fed Shean faster than you .....(viii)

could mean

. . . than I fed you,

or

. . . than you fed Shean,

and might even be combined

. . . than you typed the paper.

The dimension of comparison need not be the same as in:

"The circle is larger than the box is tall." Many ATN grammar include arcs for simple cases of comparatives, but many have not dealt with the problem in a general way.

In spite of the large ATN grammars that have been developed, a disproportionate number of arcs do not deal with the general syntactic phenomena we have enumerated. In developing a semantic grammar, one includes arcs, states, and lexical categories, tailored to fit the need of a specific subject matter. However, even in ordinary syntactic grammar there must be many specialized phrases.

There are surprisingly large number of sentences whose basic structure has been accounted for but also include one or more special combinations associated with particular words. Any speaker of a language has a large "phrasal lexicon" that associates both meanings and specialized syntactic structures with particular combinations of words. A phrase like "smart enough to know better" is peculiar to the word "enough," while "we were just about . . ." includes both "just" and "about" in structures that are not general. Thus most grammars including the one described will fall short of handling natural text and needs to be augmented with more specialized structure.

Early ATN grammars treated the building of structures independently from the use of registers and registers were often used for control purposes--as a programming device. Later uses of ATN grammar tend to encourage a structural justification for registers and to avoid adding them recklessly.

C-4.9 Form of structure

The grammar presented produces register structures, which is not a property of all ATN grammars. The final steps create and return a node containing the register table. In other versions, this is not done uniformly for all nodes. The BUILDQ actions on the arc were included to take register contents and build some kind of structure. In the early version, phrase-structure trees corresponding to the deep structures of TG were built. Other implementation have built case structures, dependency structures, etc., and a number of variants on register structures. Mostly, there has not been a tight theoretical coupling between the details of the networks and the form of resultant structures. This has been more at the whim of the grammar writer to build whatever seemed appropriate in each network.

C-5 Competence and performance

Linguists have long made a distinction between language as people actually use it (performance) and as one ideally abstract it (competence). For example, it is in the realm of competence that one knows that a reduced relative clause may be used in a NP, but it is a fact of performance that the embedding is usually performed only once and in the subject position. Competence tells us that the verb of a sentence must agree in number with the subject and sometimes with the object. There is a rule which says that a particle associated with a verb may be moved beyond the object of the verb, yet if the object is very long or complex we may consider the sentence as bad. Most system must deal with some aspects of performance which are outside the formal competence model. Thus the term "ungrammatical" becomes ambiguous in reference to the input sequence of a system--a conventionally ungrammatical sentence may be accepted by the grammar or a yet conventionally one may be rejected by the grammar.

REFERENCES

AH79 Aho, A., and J. Ullman, The theory of parsing, translation and compiling, vol. 1: Parsing. Englewood Cliffs, NJ: Prentice-Hall,1979.

√ AK75 Akmajian, A., and F. Heny, An introduction to the principles of transformational syntax. Cambridge MA: MIT Press 1975.

BA74 Bach, E., Syntactic theory. NY: Holt, Rinehart, and Winston 1974, pp.78.

BA81 Barr, A., and E.A. Feigenbaum, (Eds.), The handbook of artificial intelligence. Los Altos, CA: William Kaufman, 1981.

BA81b Bates, M., and R. Ingria, "Controlled transformational sentence generation," Proc. 19th Ann. ACL, 1981, pp.153-158.

BA78 Bates, M., "The theory and practice of augmented transition network grammars," in Natural Language Communication with Computers. L. Bolc (Ed.), NY: Springer, 1978, pp.191-260.

BO69 Bobrow, D.G., and J.B. Fraser,"An augmented state transition network analysis," in 1st. International Joint Conference on Artificial Intelligence, 1969, pp. 557-567.

BO80 Bobrow, R.J., and B.L. Weber, "Knowledge representation for syntactic/senmatic processing," in 1st. American Association for Artificial Intelligence, 1980, pp.316-323.

BO67 Bobrow, D.G., "Natural language input for a computer problem solving system," in Semantic Information Processing. M. Minsky (Ed.), Cambridge, MA: MIT Press, 1967, pp. 133-215.

BO74 Bobrow, D.,G., and B. Raphael, "New programming languages for artificial intelligence research," Computing Surveys, 6:3(1974), 153-174.

BU71 Burt, M.K., From deep to surface structure: An introduction to transformational syntax. NY: Harper and Row Publishers, 1971.

BU76 Burton, R., "Semantic grammar: An Engineering technique for constructing natural language understanding systems," BBN Report No. 3553, Bolt, Beranek, and Newman Inc., Cambridge, MA, Dec. 1976.

CA81 Carbonell, J.G., and P.J. Hayes, "Dynamic strategy selection in flexible parsing," in Proceedings 19th. Annual Assciation for Computational Linguistics, 1981, pp. 143-147.

CH81 Cherry, L., "Computer aids for writers," SIGPLAN Notices, 16, 1981, pp.61-67.

CH57 Chomsky, A.N., Syntactic structures. The Hague: Mouton & Co., 1957

CH65 Chomsky, A.N., Aspects of the theory of syntax. Cambridge, MA: MIT Press, 1965.

CO74 Colby, K.M., R. Parkinson, and W. Faught, "Pattern-matching rules for the recognition of natural language dialogue expressions," AJCL, September 1974, pp. 5-70.

EA70 Early, J., "An efficient parsing algorithm," CACM 13, 1970, pp. 94-102.

EA74 Eastman, J.F., "An N-Space Model for Visual and Verbal Concepts," Ph.D. dissertation, North Carolina State University, 1974.

FR71 Friedman, J., A computer model of transformational grammar. NY: Elsevier, 1971.

FR72 Frijda, N.H., "Recognition of emotion," in L. Berkowitz (Ed.), Advances in Experimental Social Psychology, 1972, pp. 167-223.

GR71 Griswold, R., J. Poage, and I. Polonsky, The Snobol4 programming language (2nd ed.). Englewood Cliff, NJ: Prentice-Hall, 1971.

HA81 Hayes, P.J., "A construction-specific approach to focused interaction in flexible parsing," in Proceedings 19th. Annual Association for Computational Linguistics, 1981 pp. 149-152.

HE77 Hendrix, G., "LIFER: A natural language interface facility," SIGART Newsletter, 61, 1977, pp. 25-26.

KA73 Kaplan, R.M., "A multi-processing approach to natural language," in Proceedings of the National Computer Conference, 1973, pp. 435-440.

KA75 Kaplan, R.M., "On process models for sentence analysis," in Explorations in Cognition. Donald A. Norman, and D.E. Rumelhart,and the LNR Research Group (Eds.), San Francisco: Freeman, 1975, pp.117-135.

LA75 Lakoff, G., and H. Thompson, "Introducing cognitive grammar," Proc. 1st Ann. Meeting of the Berkeley Ling. Soc., 1975, pp.295-313.

MI74 Miller, P.L., "A locally organized parser for spoken input," CACM, 17, 1974, pp.621-630.

MI81 Miller, L., G. E. Heidorn, and K. Jensen, "Text-critiquing with the EPISTLE system: an author's aid to better syntax," in Proc. of National Computer conference, 50, Arlington, VA: AFIPS Press, 1981, pp. 649-655.

NE79 Newman, W.M., and R.F. Sproull, Principles of interactive computer graphics (2nd ed.). NY: McGraw-Hill Book Co., 1979, pp.363-366.

NO69 Noble, B., Applied linear algebra. Englewood Cliff, NJ: Prentice-Hall, 1969.

PL76 Plath, W., "REQUEST: A natural language question-answering system," IBM Journ. of Res. Dev., 20, 1976, pp. 326-335.

QU67 Quillian, M.R., "Word concepts: a theory and simulation of some basic semantic capabilities," Behavioral Science, 12, 1967, pp.410-430.

QU69 Quillian, M.R., "The teachable language comprehender: A simulation program and theory of language," CACM, 12, 1969, pp.459-476.

RI83 Rich, E., Artificial intelligence. NY: McGraw-Hill Book Co., 1983, pp.201-242.

SA81 Sager, N., Natural language information processing. Reading, MA: Addison-Wesley, 1981.

SC73 Schank, R., "Identification of conceptualizations underlying natural language," in R. Schank and M. Colby, (Eds.), Computer Models of Thought and Language, San Francisco: Freeman,1973, pp.187-248.

SI73 Simmons, R.F., Semantic Networks: Their computation and use for the understanding of english sentences," in R. Schank and M. Colby, (Eds.), Computer Models of Thought and Language, San Francisco: Freeman, 1973, pp. 63-113.

SI72 Simmons, R.F., and J. Slocum, "Generating english discourse from semantic networks," CACM 15, 1972, pp.891-905.

TH69 Thompson, F., P. Lockemann,and B. Dostert, "REL: a rapidly extensible language system," Proc. 24th National Conf. of the ACM, 1969, pp.399.

TH81 Thompson, H., "Chart parsing and rule schemata in PSG," 19th Ann. ACL, 1981, pp.167-172.

VA75 Valiant, L., "General context free recognition in less than cubic time," J. Computer and Systems Sciences 10:2(1975) pp.308-315.

WA78 Warner, E., and M. Maratsos, "An ATN approach to comprehension," in Linguistic Theory and Psychological Reality. M. Halle, J. Bresnan, and G. Miller (Eds.), Cambridge, MA: MIT Press, 1978, pp.119-161.

WI72 Winograd, T., Understanding natural language. NY: Academic Press, 1972.

WI83 Winograd, T., Language as a cognitive process. Vol.1: Syntax Reading,MA: Addison-Wesley Publ. Co., 1983, Syntax.

WI79 Winston, P.H., Artificial intelligence. Reading,MA: Addison-Wesley Publ. Co., 1979.

WI79b Winston, P.H., "Learning by creating and justifying transfer frames," in Artificial Intelligence: An MIT Perspective. P.H. Winston, R.H. Brown (eds.), Cambridge, MA: MIT Press, 1979.

WO70 Woods, W.A., "Transition network grammars for natural language analysis," CACM, 13, 1970, pp.591-666.

WO73 Woods, W.A., "Progress in natural language understanding: An application to lunar geology," AFIPS Conference Proceedings, 42, 1973, pp.441-450.

BIOGRAPHICAL SKETCH

Alade Olabambo Tokuta was born in Bida, Nigeria, to Jibril and Safuratu Tokuta. He obtained his B.S.E.E. and M.S.E.E. degrees in electrical engineering, from Duke University, Durham, North Carolina.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John Staudhammer

John Staudhammer, Chairman
Professor of Electrical Engineering
and Computer Science

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Eugene R. Chenette

Eugene Chenette, Cochairman
Professor of Elecrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jack Smith

Jack Smith
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Leslie Oliver
Associate Professor of Computer
and Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Delbert Tesar

Delbert Tesar
Graduate Research Professor of
Mechanical Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School, and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philisophy

April 1984

Herbert A. Bevis

Dean, College of Engineering

Dean for Graduate Studies and
Research